Rapport Projet 2018

Sujet : Développement D'un Jeu GO GAME

OUTIDRARINE Mohamed

harrymomment@gmail.com

Encadrement : Mlle Mariam CHERRABI

SOMMAIRE

I. Introduction

Ce projet a été effectué dans le cadre du module technique de programmations dans une durée de 45 jours pendant laquelle nous étions encadrés par le professeur **Mme M.Cherrabi** que nous remercions pour nous avoir donnée cette opportunité de travailler sur un projet réel qui a pris comme sujet cette année le **jeu GO.**

D'après le cahier de charges nous sommes amenés à programmer le jeu GO avec le langage C et produire par la suite deux versions du jeu, une version console et une version graphique implémenté par la bibliothèque graphique SDL.

Durant les 45 jours de travaille qu'on a effectué pour travaillé ce projet nous avons rencontré des difficultés que nous avons pu surmonter, nous avons fait des choix de stratégie pour aboutir finalement à ce produit.

Dans ce rapport on va présenter tous les difficultés qu'on a trouvées, et on va expliquer comment on a fait pour surmonter ces difficultés. On va justifier des choix de stratégies pour lesquels on a opté et les choix qu'on a eu et parmi lesquels on a choisi ces stratégies, on va expliquer les raisons qui nous ont poussé à faire ces choix en détaillant les limitations et les points fort de chaque proposition qu'on s'est proposé.

II. A propos du jeu

1- Présentation du jeu :

Le go est un jeu de plateau originaire de Chine. Il oppose deux adversaires qui placent à tour de rôle des pierres, respectivement noires et blanches, sur les intersections d'un tablier quadrillé appelé goban. Le but est de contrôler le plan de jeu en y construisant des « territoires ». Les pierres encerclées deviennent des « prisonniers », le gagnant étant le joueur ayant totalisé le plus de territoires et de prisonniers. -- wikipedia

2- Règles du jeu

➢ Chaînes et libertés

Les pierres de même couleur qui sont directement adjacentes en suivant les lignes de la grille (une pierre a donc au plus quatre voisines) sont dites *connectées* et forment une **chaîne**. On appelle *libertés* les intersections vides immédiatement adjacentes à une chaîne de pierres.

➢ Capture

Une pierre isolée ou plus généralement une chaine qui ne possède plus qu'une seule *liberté* est dite en atari. Si la chaîne perd cette dernière *liberté*, elle est *capturée*. La chaine complète est retirée du *goban* et ajoutée au tas de *prisonniers* du joueur adverse.

➢ Coups interdits

○ *Suicide*

Lorsqu'on joue une pierre, on examine d'abord les libertés des groupes adverses à son voisinage. Si certains d'entre eux n'ont plus de libertés, on les enlève et on les compte comme prisonniers. Si aucun groupe adverse n'a ainsi été capturé, on étudie les libertés du groupe auquel appartient désormais la pierre qui vient d'être posée (éventuellement, elle seule). Si ce groupe n'a pas de libertés, le coup est un *suicide* : il est interdit.

○ *Ko*

Pour éviter qu'une situation ne se répète à l'infini, la règle du ko (un mot japonais signifiant éternité) interdit de jouer un coup qui ramènerait le jeu dans une position déjà vue dans le courant de la partie.

➢ Fin de partie

Si aucun des joueurs n'a abandonné, la partie se termine après que les deux joueurs ont passé consécutivement. On comptabilise alors les points de chacun. Celui qui possède le plus de points gagne.

➢ Komi

Après le décompte, il ne faut pas oublier d'ajouter les points du komi au total du joueur blanc.
La valeur du komi varie suivant les pays. Il est actuellement de 6,5 points au Japon et en Corée, et de 7,5 en Chine et en France.

➢ Territoire

Un territoire est un ensemble de une ou plusieurs intersections inoccupées voisines de proche en proche, délimitées par des pierres de même couleur.

➢ Décompte final

La partie s'arrête lorsque les deux joueurs passent consécutivement. On compte alors les points. Chaque intersection du
Territoire d'un joueur lui rapporte un point, ainsi que chacune de ses pierres encore présentes sur le goban.
Par ailleurs, commencer est un avantage pour Noir. Aussi, dans une partie à égalité, Blanc reçoit en échange des points de
Compensation, appelles komi. Le komi est habituellement de 7 points et demi (le demi-point sert à éviter les parties nulles).
Le gagnant est celui qui a le plus de points.

II. A propos du projet

1- Travail demandé

Le projet consiste à développer le jeu GO en utilisant le langage C, le jeu à créer doit implémenter la possibilité d'avoir 2 joueurs Hommes, et aussi de pouvoir jouer contre la machine.
le projet doit être réalisé en deux versions, une version console, et une version graphique implémenté par la bibliothèque graphique SDL du langage C.
le travail final doit respecter les règles conventionnels du jeu (capture, territoires, coups interdits…).

3- Les gros travaux réalisés dans le projet :

Dans notre travail nous avons réalisé :

- ✓ la version console du jeu
- ✓ la version graphique implémentée par la bibliothèque SDL
- ✓ le fonctionnement de la capture des pierres n'ayant point de degrés de libertés.
- ✓ le fonctionnement du calcul du degré de libertés des pierres et des groupes de pierres.
- ✓ le fonctionnement du calcul des territoires occupés par des joueurs.

- ✓ le fonctionnement nécessaire pour interdire tous les coups interdits comme le suicide et le KO.
- ✓ Interface IHM pour les deux versions avec un fonctionnement élégant pour les deux versions SDL et console.
- ✓ Faire en sorte que le jeu produit soit intègre et respect le fonctionnement naturelle du jeu GO et ses règles conventionnels.

4- Contraintes et problèmes générales rencontrés dans le projet :

Par ordre de complexité on va citer les grosses contraintes rencontrées.

- **contrainte de temps :** malgré que nous avons été conscient que le temps est très pressé dès la réception du cahier de charge, et malgré que nous avons commencé de manœuvrer dès la réception du cahier de charges, cette problématique nous fus inévitable, cependant nous avons veuillez à réaliser la maximum des travaux possible.

- **contrainte d'optimisation du jeu :** cette contrainte nous a pris beaucoup de temps surtout dans la version SDL, parce que dans un jeu go il y'a en général beaucoup plus de traitement à faire après chaque coup nous allons en citer (éviter les coups interdits, faire disparaitre du goban tous les pierres capturés, calcul des territoires…) ce qui vous a poussé à déployer un effort supplémentaire pour optimiser le produit et final et le rendre un peut plus léger.

- **comprendre le jeu et ces règles :** ce qui rend le jeu go très complexe, c'est le fait que les règles du jeu varient d'un pays à l'autre, par exemples il y'a des règles Chinois, des règles Japonais et des règles françaises. Dans un certains moment lorsqu'on est amené à comprendre le fonctionnement du calcul des territoires, nous avons subie on grande confusions du au plusieurs variantes des règles… et finalement nous avons décidé d'implémenter les règles françaises du jeu (PieceJointe2) après avoir discuter ce problèmes avec des membres d'un réseau social qui s'appel REDDIT, où nous avons rencontré des gens qui ont déjà travaillé sur le jeu go et puis nous avons sollicité leurs expérience pour trouver des pistes que nous avons exploiter par la suite et qui sont aboutie à des solutions et des idées dont on a beaucoup bénéficier.

5- Outils et supports utilisés

➢ La bibliothèque SDL du langage c

C'est une bibliothèque logicielle libre. Son API est utilisée pour créer des applications multimédias en deux dimensions pouvant comprendre du son comme les jeux vidéo, les démos graphiques, les émulateurs, etc. Sa portabilité sur la plupart des plateformes et sa licence zlib, très permissive contribuent à son succès. –wikipedia

Nous avons pris une durée de 2 jours pour bien saisir et apprendre le fonctionnement de la bibliothèque SDL, nous avons suivi la série des leçons proposées par lazyfoo.net qui sont beaucoup détaillé et claires.
Nous avons également appris bien manipuler les événements qui s produisent durant le jeu, c'est à dire l'ensemble des événements qui vont conduire le fonctionnement du jeu tel que les clicks et les mouvements de la souris.

Nous avons utilisé la bibliothèque extension SDL2_image pour pouvoir intégrer des images de différents formats (JPG, PNG ..ect) et non las seulement Bitmap.

➢ Photoshop :

Pour la création des graphiques, on a utilisé le logiciel **Adobe Photoshop** afin de créer un jeu bien conçu et un design qui plait à l'oeil. Vous trouvez tous les images constituant notre jeu dans **le dossier Images.** Et on va aussi inclure les fichiers Photoshop **.psd**

➢ ASCII art :

Ascii art est une site web qui génère automatiquement des textes décoré avec des caractères ASCII, nous avons utiliser cet outis pour la version console du jeu.

➢ La bibliothèque <windows.h> :

Cette bibliothèque contient des fonctions très intéressantes qui permet d'améliorer le graphique de la console et avoir un jeu console qui est mieux décoré et plaisant à l'oie. Voici quelques fonctionnalités que permet les fonctions que contient cette bibliothèque :

✓ Changer les couleurs du text et d'arrière plan :

```
void color(int t, int f)
{
HANDLE H = GetStdHandle(STD_OUTPUT_HANDLE);
SetConsoleTextAttribute(H, f * 16 + t);
}
```

Cette fonction qu'on a construit à l'aide de quelques fonctions que contient la cette bibliothèque permet de changer la couleur du text et de l'arrière plan. Les couleurs sont représentés par des entiers la figure suivante montre l'entier équivalent à chaque couleur.

Colour codes available:

Code (Hex)	Color
0	black
1	blue
2	green
3	cyan
4	red
5	magenta
6	yellow/brown
7	white
8	gray
9	bright blue
A	bright green
B	bright cyan
C	bright red
D	bright magenta
E	bright yellow
F	white

✓ Nettoyer l'a console :

Avec un simple appel à la fonction au-dessous on pourra nettoyer l'écran pour la remplir avec le contenue généré pour le coup de role suivant.

```
system("@cls||clear");
```

III. Navigation dans le jeu.

1- La version graphique SDL

On va naviguer maintenant dans les différents menus du jeu pour découvrir son ergonomie et son fonctionnement.

➢ Page d'accueil

La page d'accueil offre 4 choix :

- ✓ **Play** : permet d'accéder aux modes du jeu
- ✓ **Rules** : permet une consultation des règles principales du jeu
- ✓ **options** : ce bouton a été conceptionné dans la phase du conceptions afin de donner la main aux utilisateurs d'activer ou désactiver le son, malheureusement on a pas pu terminer cette cet partie à cause du contrainte de temps, mais on l'a laissé ouverte dans le code pour une éventuelle
- ✓ **exit** : pour fermer le jeu définitivement.

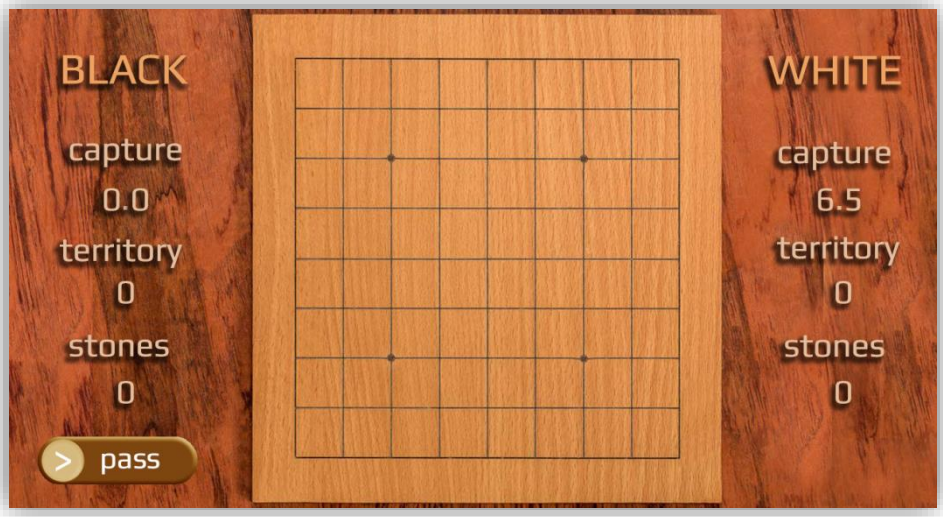
➢ Les modes du jeu :

Lorsqu'on click sur play dans le menu d'accueil on accède directement aux modes du jeu, on a réalisé deux modes comme exigé par le cahier de charges.

- ✓ **Mode 2players :** ici pour donner la main à deux joueurs humains pour jouer.
- ✓ **Mode 1player :** pour donner la main à un seul joueur pou qu'il puisse jouer contre le CPU.
- ✓ **Return :** pour retourner au menu principal

➢ Joueurs humain vs humain (2 players) :

Lorsqu'on appui sur le bouton 2players de menu modes du jeu on accède directement au goban et la partie commence directement.



➢ Les composantes du goban :

- ✓ **Bouton pass :** qui indique dans le bas le joueur qui a le tour de rôle, et qui permet aussi aux joueurs d'abandonner son tour de rôle et de passer la main à l'autre joueur pour jouer. Ce bouton change place a chaque tour de rôle et se met dans le coté du joueur qui a la main pour jouer.

- ✓ **Les informations sur chaque joueur :** on a les informations concernant le joueur noir du coté gauche du goban, les informations sur le joueur blanc du coté droit du goban.

- ✓ **Capture :** cette rubrique indique le nombre de pierres capturés par chaque joueur

- ✓ **Terriotry :** ici on indique le nombre de points apportés à chaque joueur grâce à la constitution de son territoire.
- ✓ **Stones :** dans cette rubrique on indique le nombre de pierres existante dans le goban pour chaque joueur.

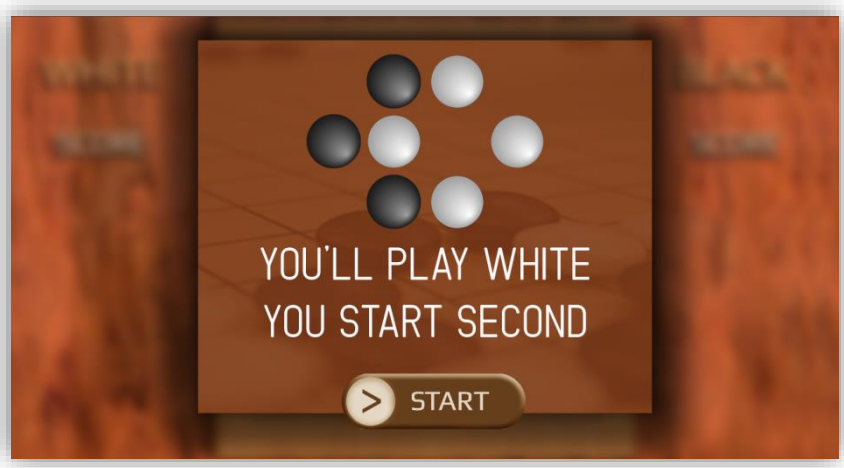
- ➢ **Remarque :** Les trois types de points indiqués au-dessus sont importants pour le calcul du score de chaque joueur et détermination du gagnant.

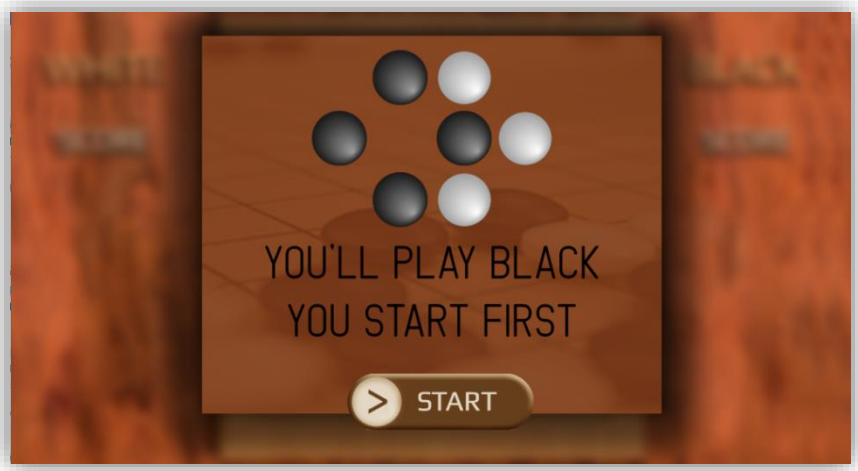
➢ Joueur humain vs machine

Quand on clique sur le bouton play on obtient directement les différents niveaux de difficultés pour le jouer contre la machine.

➢ Tirage au sort

Lorsque l'utilisateur choisie le mode humain vs machine, un tirage au sort est nécessaire pour décider celui qui va commencer son tour de rôle le premier, pour réaliser ce tirage au sort voici le résultat qu'on a produit.





On a eu l'idée de réaliser un tirage au sort qui est propre au jeu go et qui est significatif élégant et convenant au jeu GO, c'est pour cela nous avons pris le même principe du pile ou face, mais avec une version GO-GAME.
L'utilisateur doit attendre des coups successifs du famous KO, qui est un cas très particulier dans le jeu GO, et qui est considéré comme un coup interdit.

Les coups successifs du KO termineront avec une situation qui stipule le joueur qui aura la main pour commencer le premier. En gros le l'utilisateur prendra la

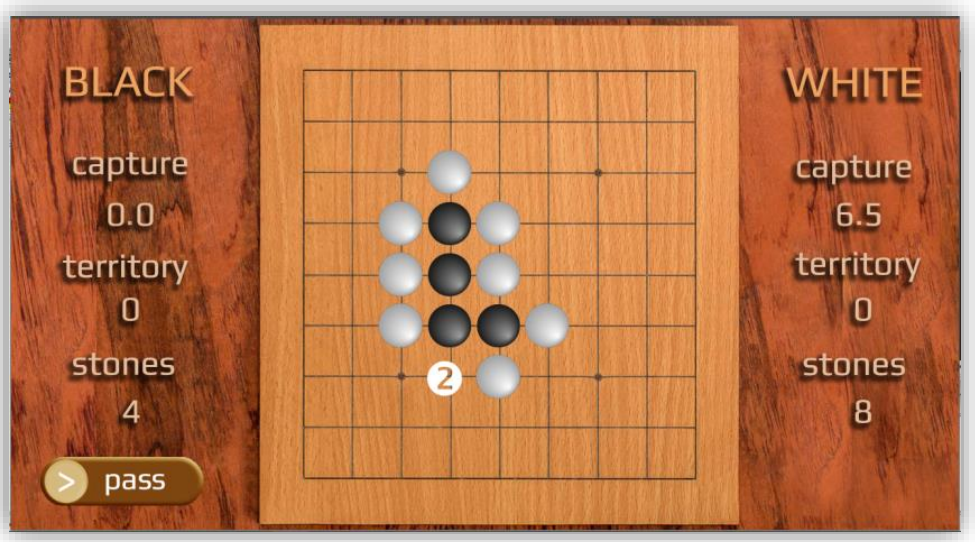
pierre qui va rester dans la situation finale, avec la convention qui stipule que le noir commence toujours le premier.

➢ Sur le GOBAN :

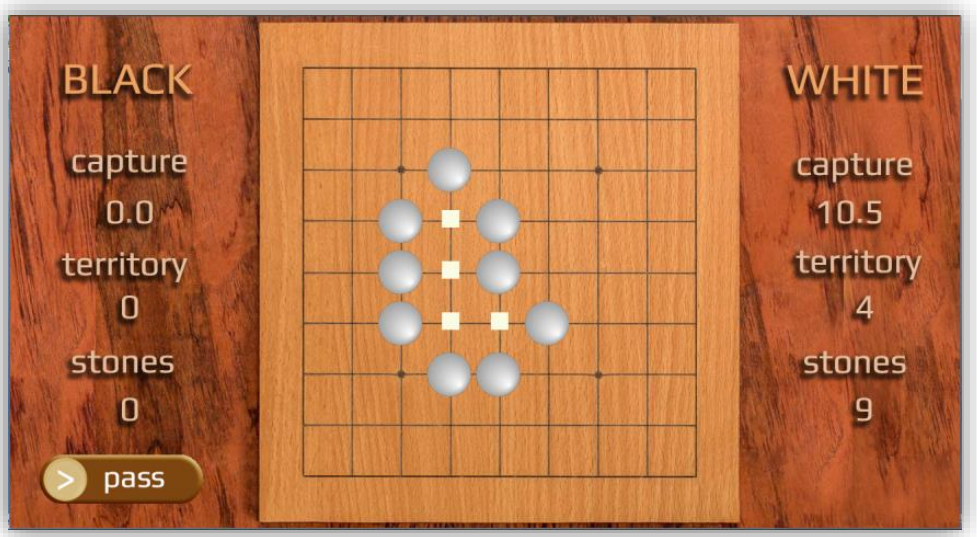
Une fois l'utilisateur sur le goban, entrain de jouer, le programme suit les coups produits par les utilisateurs (ou l'utilisateur et le cpu éventuellement) et puis produit les actions nécessaires tel que la capture est les coups interdits ainsi que les territoires occupées.
Maintenant on va explorer des exemples de situations dans lesquels le programmes produit les réactions nécessaires suivent les coup de l'utilisateur.

➢ Capture

Prenons par exemple cette situation



Lorsqu'on place une pierre blanche sur l'intersection qui contient le nombre 2, le groupe des pierres noires doivent être capturés et par la suite ils doivent disparaitre du goban.

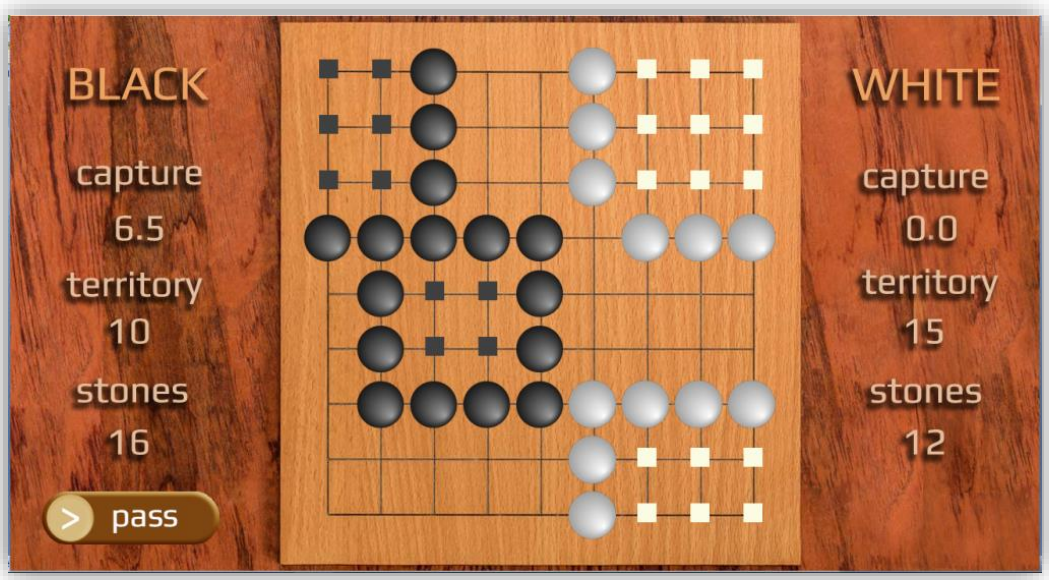


C'est effectivement le cas, et c'est ce qui ce passé effectivement dans le programme, la figure sur dessus illustre le cas.

- ✓ **Remarque :** la capturation des pieres individuelles va de la même façon que la capturation des groupes de pierres. Dans le code qu'on a implémenté une pierre individuelle est considéré un cas particulier du groupe de pierres, en tt cas une pierre ou groupe de pierres n'ayant pas de degrés de libertés serons capturés.

➢ Territoire

Quand un territoire est devenu occupé par un joueur, le programme doit l'illustrer sur le goban. Voici quelques exemples des territoires occupés et illustrés sur le goban.



✓ **Remarque** : les règles de constitution des territoires nous ont pris beaucoup de temps de recherche, surtout qu'il ya beaucoup de confusions entre des règles japonais et des règles chinois, après beaucoup de recherche on a choisi d'implémenter les règles françaises.

1- **A propos d'une confusion qui est trop propagé :** pendant nos recherches sur les règles du KO nous nous sommes demandé souvent la question suivante, dans un territoire déjà constitué, si le joueur adverse a posé une pierre dans ce territoire est-ce que ce territoire sera encore considéré comme territoire et sera calculé par la suite dans le score total du joueur.

pour trouver une repense à cette question nous avons entamé beaucoup de discussions et beaucoup de recherche, et finalement nous avons aboutie au résultat suivant :

d'après les règles français du jeu (PieceJointe2 : Un territoire est un ensemble de une ou plusieurs intersections inoccupées voisines de proche en proche, délimitées par des pierres de même couleur.)
la problématique se prose lorsque un joueur adverse vient poser une pierre sur ce territoire sera il considéré encore territoire ?

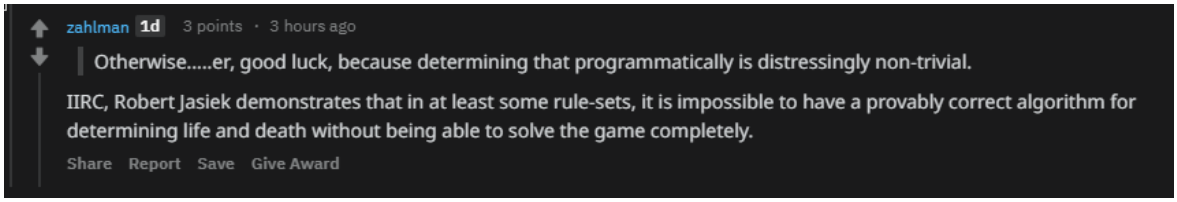
➔ Voici la réponse : non, il ne sera pas considéré comme territoire, pacque sinon le joueur adverse peut constituer des eux à l'intérieure

de ce territoire et le fait de le considérer encore territoire dans cette situation est absurde.

Cependant le joueur n'a pas intérêt à posé une pierre dans le territoire du joueur adverse parce qu'elle sera captura facilement, la capturation est en plus assuré.

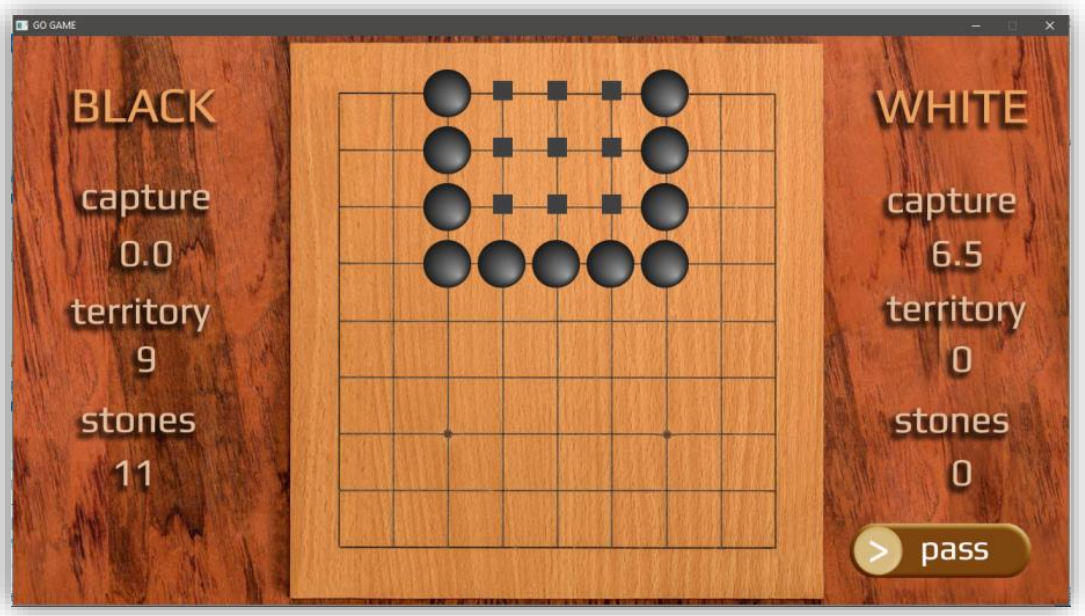
Normalement dans le jeu les deux joueurs font un compromis, si il s'avère qu'un territoire appartient à un joueur, ils se mettent d'accord de le considérer comme territoire acquis par le joueur pour gagner du temps dans le jeu.

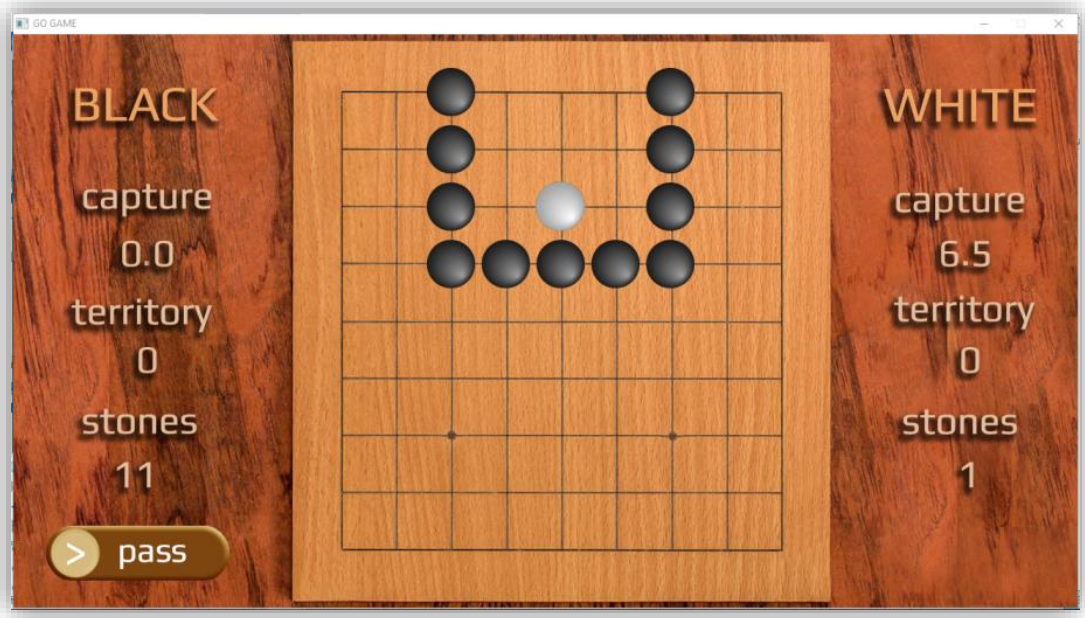
Nous avons aussi posé des question autour de cette problématique dans le réseau REDDIT et nous avons obtenu des réponses intéressantes, voici une :



D'après cet démonstration de Rober Jasiek, il est impossible de trouver un programme qui va nous dire si un territoire va toujours rester établie durant le jeu, d'où la légitimité du faite de ne pas considéré un territoire comme territoire lorsqu'un joueur adverse a posé son pierre dans un territoire.

Revenant maintenant à notre jeu, on a implémenté les règles du calcul des territoires de tels sort qu'ils vérifient les conditions motionnées au-dessus, c'est-à-dire lorsque le joueur adverse pose une pierre dans un territoire il faut plus le considérer territoire.





➢ Les coups interdits :

○ suicide :

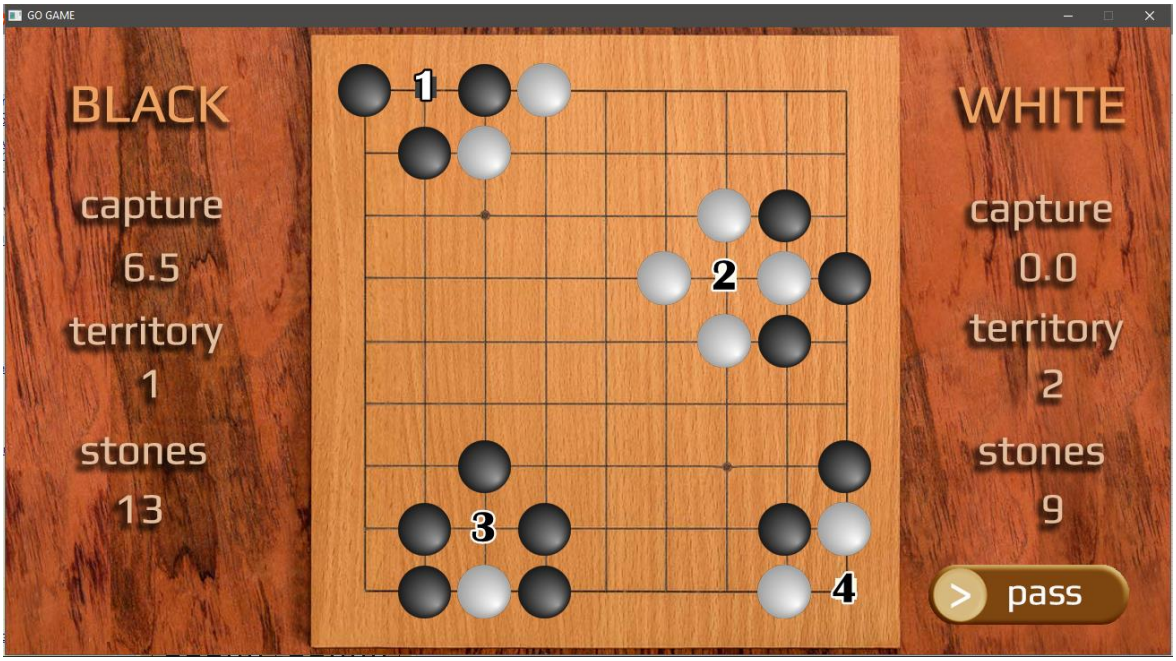
Une pierre ne peut pas être posé sur une intersection si il aura une degré de liberté nulle. Nous avons respecté cette règle dans notre jeu avec ses deux versions, effectivement lorsqu'on essaie de poser une pierre sur une

intersection sur lequel elle aura une degré de liberté nulle il nous empêche.

Un coup suicidaire peut facilement être identifié, il se produit lorsqu'on pose une pierre avec un groupe de pierres de telle façon à ce que ce groupe de pierres ait une degré de liberté nulle. Notre programme empêche ce cas de se produire (pace que sinon ce sera une limitation pour notre jeu).

○ *Le coup KO :*

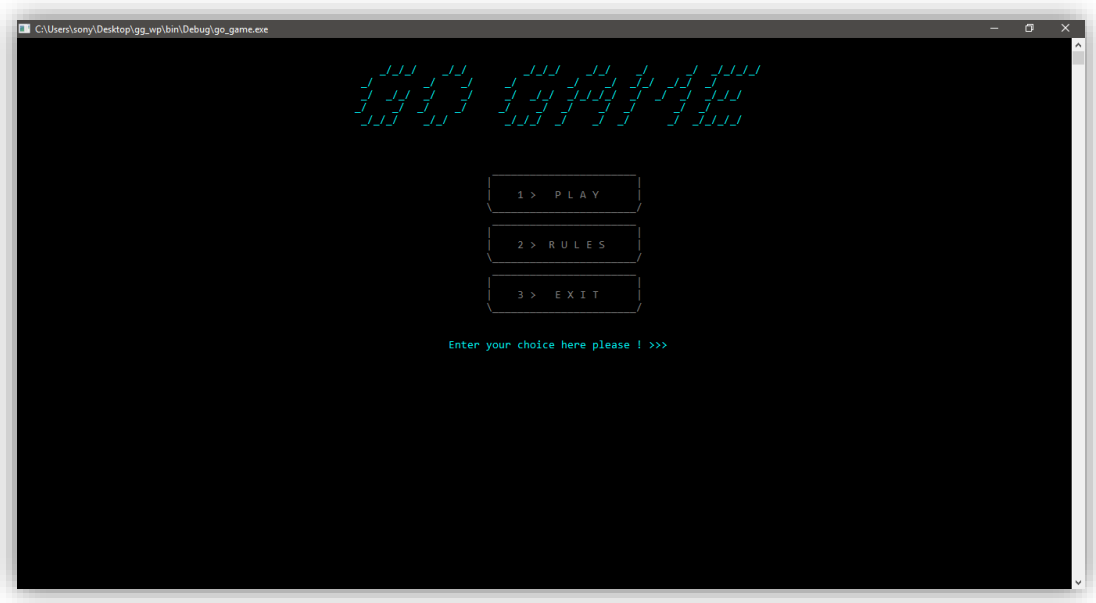
Nous avons déjà expliqué le coup KO dans la partie règles du jeu, ce dernier est considéré un coup interdit dans le jeu, nous avons respecté cette règle, voici quelques cas de figures des coups de KO :



Dans notre programme nous avons implémenté une fonction qui évite que ses cas se produisent, et tous les autres cas si y'en a.

6- La version console du jeu :

➢ 1-menu d'accueil :

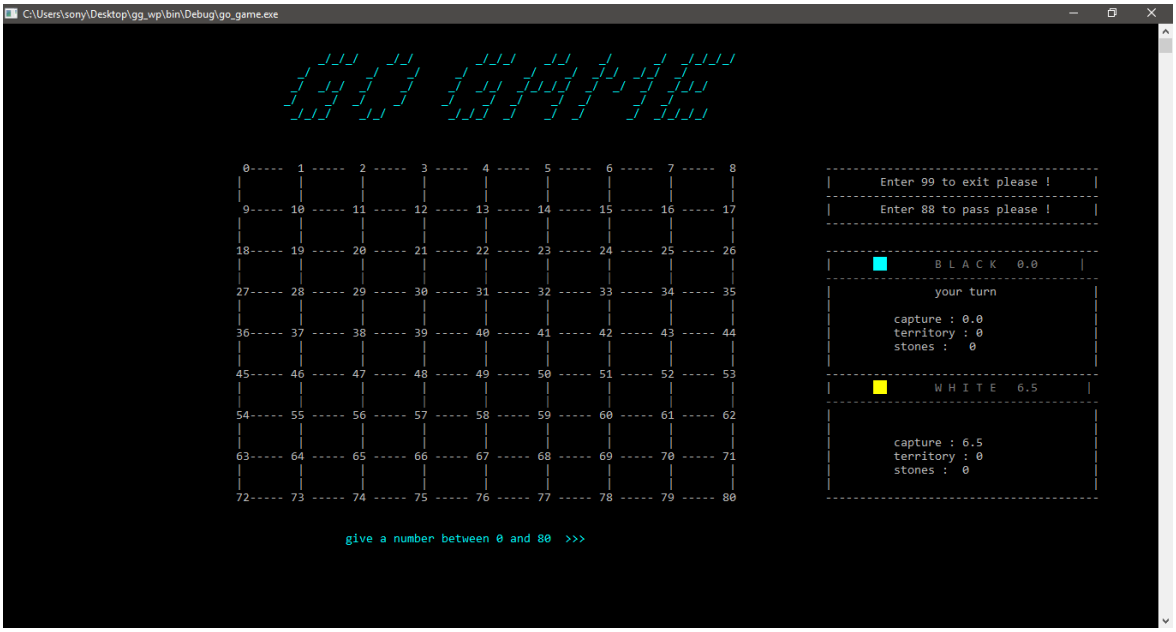


La page d'accueil de la version console possède le même squelette que la page d'accueil de la version SDL

➢ 2- les modes du jeu :

Les mêmes modes qu'on a déjà cités pour la version SDL.

➢ 2- Le goban :

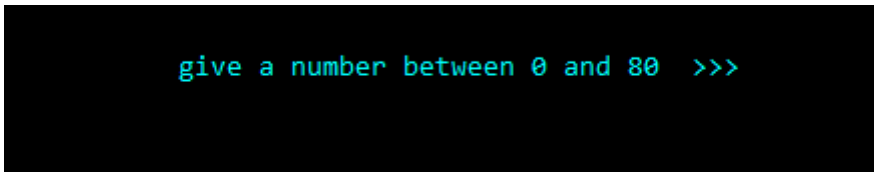


Voici le goban de la version console du jeu, nous avons jugé judicieux d'implémenter une conception différente du jeu original, nous avons abandonnée le concept de l'abscisse et de l'ordonnée des intersections, et nous avons garder un goban qui contient des intersections numéroté, pour poser une pierre sur une intersection l'utilisateur n'a besoin de saisir qu'un seul nombre au lieu de deux nombres, ce qui laisse déjà une expérience de l'utilisateur fluide et lisse.

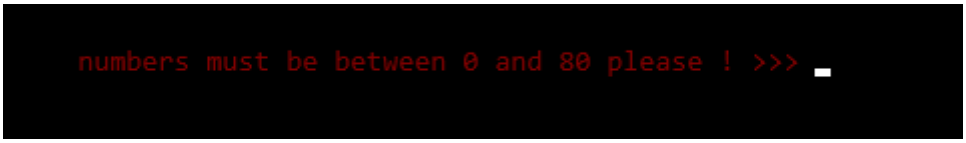
Cette conception différence nous a aussi servie dans le coté codage, parce que grâce à cette conception nous avons travaillé qu'avec des tableaux d'une seul dimension, ce qui permet de parcourir les tableaux avec un seul indice au lieu de 2 indices, comme ca le tableau est manipulable facilement.

➢ Les messages l'orientations

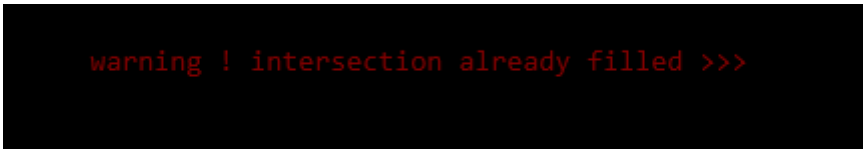
Sur le goban on a implémenté des messages qui orientent l'utilisateur en cas d'erreurs, pour l'orienter et lui montrer ce qui ce qui cloche.
voici quelques exemples des messages d'orientation des utilisateurs :
Par défaut on indique à l'utilisateur de saisir un entier qui est compris entre 0 et 80



Si l'utilisateur a donné une valeur qui n'est pas comprise entre 0 et 80 on lui affiche un message d'erreur qui indique la source de l'erreur.



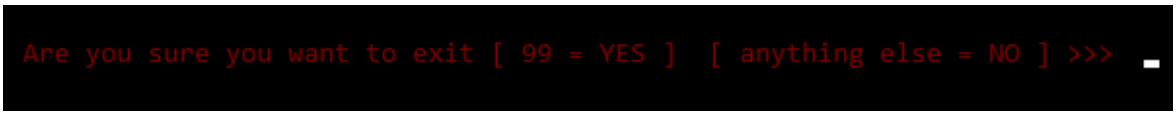
Si l'utiliateur a saisie la valeur d'une intersection qui est déjà remplie, on lui affiche le message suivant.



Parmi les coups interdits il ya le famous coup KO, si l'utilisateur a donnée une valeur qui amènera à la situation du KO on lui affiche le message suivant.

Un autre coup interdit, quand l'utilisateur entre la valeur d'une intersection dont le degré de liberté est nul ou bien une valeur qui va amena à la situation d'un suicide on lui affiche le message suivant.

Sur le goban on a implémenté une possibilité de quitter le jeu en saisissant le nombre 99, (et 88 pour un pass) si l'utilisateur demande de quitter une certaine partie on lui demande d'abord de confirmer qu'il veut quitter.



IV. Conception du jeu :

Dans cette phase très importante nous allons voir comment on a répondu à aux questions qui tourne autour de la conception et codage du jeu :

1- Représentation et stockage des informations collectés :

D'abord la question était auparavant, est-ce qu'on doit stocker les informations qui comportent les coups déjà jouées ou bien on n'a besoin que de les afficher. Effectivement la question était légitime car parfois on n'a pas besoin de stocker certaines informations et on se contente juste de les afficher dès le tour en question, mais dans le jeu GO on aura besoin de calculer les degrés de liberté qui peuvent être changés après chaque tour de rôle, et parfois on doit même éliminer les pierres dont les degrés de libertés sont nuls.

- ✓ **Conclusion :** on doit stocker les informations qui se collectent durant les tours de rôles

Maintenant qu'on a conclu que le stockage des informations collectés et légitime, une autre question se pose, c'est la suivante

7- Méthode de représentation et stockage des données collectés :

On va maintenant parler de quelques propositions qu'on s'est proposé durant la phase de conception et voir justifier le choix qu'on a effectué.

➢ Structure de données liste chainé : (stratégie éliminé)

La première conception qu'on s'est proposé, c'est de stocker l'ensemble des informations dans une liste chaine, ce choix s'est avéré inefficace par la suite pour les raisons suivantes :

les listes chainés sont une structure de données linéaire qui favorise le mise à jour et la modification, c'est-à-dire que son efficacité réside dans l'aspect où elle

est facile d'y ajouter er enlever des informations, efficace dans un tableau si on veut ajouter une information au milieu il faudra faire une translation de toutes les informations qui suivent pour ajouter la nouvelle.
Ceci étant dit est les points forts d'une liste chainé, or une telle structure de donnée possède une limitation qui réside dans le faite qu'elle ne favorise pas l'accès rapide à une information, en effet pour accéder à une information dans l'indice N ème, il faut parcourir les indice une à une jusqu'à arriver à l'indice N ème. Et dans le jeu GO on aura souvent besoin de ces deux opérations pour ajouter une pierre qui vient d'être posé et pour calculer les degrés de libertés des pierres.

- ✓ **Conclusion :** le choix d'utiliser une liste chainée pour représenter les informations collectées n'est pas très efficace.

➢ Tableau de deux dimensions (stratégie éliminé) :

Le deuxième réflexion qu'on a eu à propos de la méthode de stockage des informations est de stocker les informations dans une structure d'un tableau de deux dimensions.
on a déjà discuter dans la partie –navigation dans le jeu- une idée qui nous apparait très optimale qui consiste en l'utilisation d'une seule indice pour donner le choix aux joueurs de choisir une intersection, et on a dit que ce choix nous amènera aussi à utiliser des tableaux d'une seule dimension pour stocker les informations collecté ce qui est un avantage de plus.

- ✓ **Conclusion** : un tableau de deux dimensions n'est pas le choix le plus optimale pour représenter les informations collectées

➢ Tableau de à une dimension (stratégie prise) :

Le choix qu'on a déjà fait concernant utiliser des entier de 0 à 80 pour adresser les intersections sur le goban pour les joueur, et que nous avons par ailleurs déjà discuter dans la partie concernant navigation dans le jeu, nous a donnée cet avantage de représenter tous les données avec des tableau d'une seul dimension.

- ✓ **Conclusion :** on va prendre tous les informations qu'on collecte durant une partie et on va les stocker dans des tableaux d'une dimension.

8- Représenter le goban sur la version console du jeu :

- Commençons par les évidences suivantes :
  Chaque intersection dans le jeu go peut être représentée en états :

➢ **Intersection vide :** lorsqu'une intersection est vide elle doit contenir un nombre qui est dans l'intervalle [0,80] et ceci pour donner au joueur le choix de la remplir lorsqu'il veut.

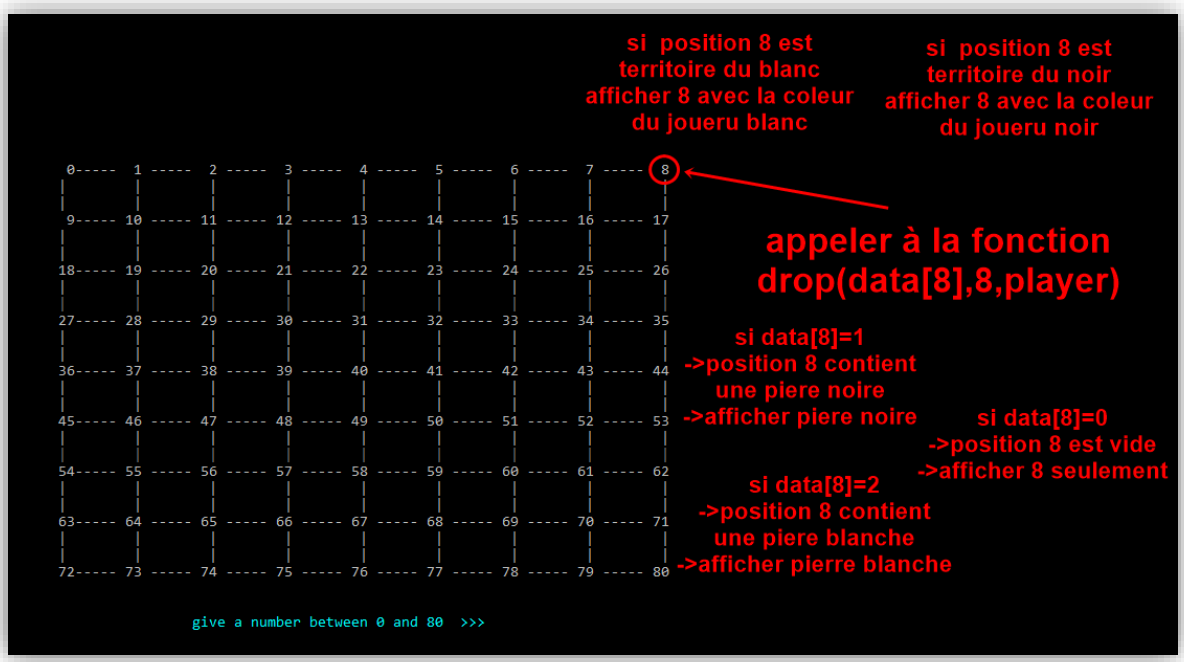
➢ **Intersection remplie par le joueur noir :** elle doit contenir une pierre noire ou bien une autre couleur qui jouera le role de la couleur noir.

➢ **Intersection remplie par le joueur blanc :** lorsqu'une intersection est remplie par un joueur blanc elle doit être remplie par une couleur qui représentera le blanc.

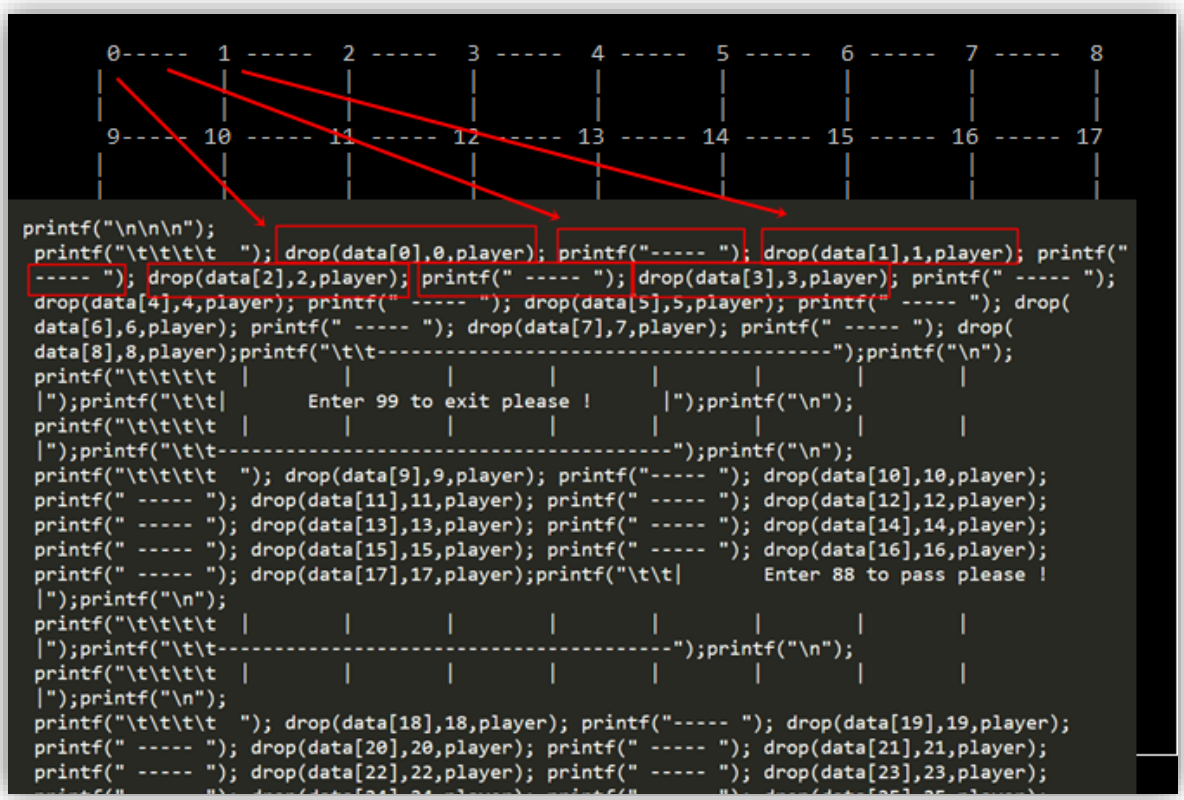
- une autre contrainte est omniprésente, il s'agit du faite qu'une intersection peut être remplie est devenir vide par la suite si la pierre en question a été capturé.

- une contrainte qui est aussi omniprésente, lorsqu'une intersection est vide mais appartient aux territoires d'un joueur, elle doit afficher quelque chose qui va être significatif, et qui signifiera qui cette intersection en question appartient au territoire de ce joueur, la solution apporté pour répondre à cette question sera discuté par la suite.

✓ **Solution :** Pour satisfaire toutes ces contraintes on a choisi d'implémenter une fonction qui va s'occuper de faire le choix de ce qui sera afficher dans les intersections, ce choix prendra compte de tous les contraintes qu'on vient de citer pour prendre.



Dans notre code on à implémenter dans chaque intersection l'appel d'une fonction qui fera les testes illustré dans la figure représenté si dessus
Pour effectuer cette opération voici une aperçu du code illustrant comment on a procéder



Un tell code a l'apparence délicat et difficile à implémenter cependant, après avoir implémenter la première ligne on pourra facilement obtenir les lignes suivantes avec des simples copie coller.

V. Programmation des règles du jeu

1- Programmation de la capture des pierres :

La capture est dans le jeu go subie des règles très particulière on a déjà expliqué comment capturer une pierre dans la partie règles du jeu, reste à voire comment on a implémenter une fonction qui s'occupera de cette tache.
On va d'abord exposer les contraintes qui nous ont amené à faire des choix stratégiques pour programmer la capture de pierres.

○ *Choisir une stratégie pour programmer la capture des pierres*

Dans le début nous étions amenés à choisir entre deux stratégies :

- programmer une fonction qui calcul les degrés de liberté de chaque pierre est chaque groupe de pierres, sur la base de cette fonction sera basé une autre fonction qui va servir à capturer les pierres dont les degrés de libertés sont nulles.
- programmer une fonction qui va parcourir le goban d'une certaines façons qu'on aura programmé, et qui va calculer les degrés de libertés de chaque pierres, cette fonction fera la capture des pierres sans attribution de degrés de libertés

Après un brain storming qu'on a effectué nous avons pris une décision, voici la décision et voici les contraintes qui nous ont poussés à choisir cette stratégie :

Nous avons jugé judicieux de choisir la première stratégie pour les raisons suivantes :

- On aura besoins des degrés de libertés des pierres et des groupes de pierres lorsqu'on a abordé la parie concernant l'intelligence artificiel, cette AI qu'on u eu l'intention de programmer se basera certainement sur les degrés de libertés pour choisir où positionner.

- Si nous avons opté pour la deuxième stratégie il y'aura une redondance parceque on va répéter un nombre important de certaines instructions plusieurs fois, notamment dans la partie intelligence artificielle.

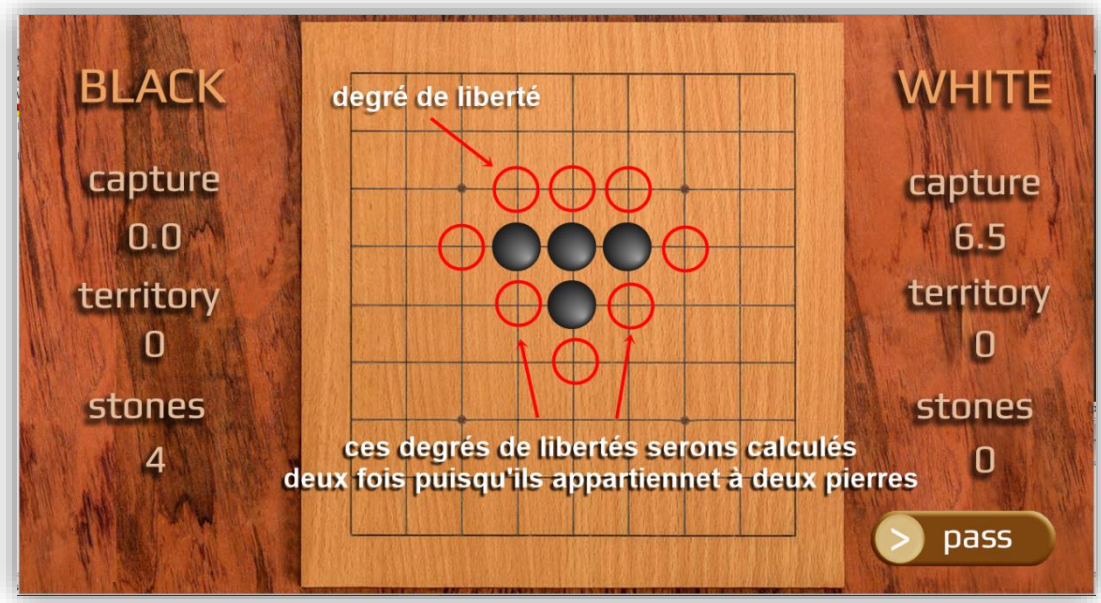
- ✓ **Conclusion :** pour aborder la capture des pierres on va programmer une fonction qui va calculer les degrés de libertés des pierres et des groupes de pierres, sur la base de cette fonction sera élaborée une autre fonction qui fera la capture.

Maintenant qu'on a opté pour une stratégie globale, venons découvrir les détails, dans la suite on va répondre à la question suivante, comment on va programmer la fonction qui s'occupera du calcul des degrés de libertés.

○ *Opter pour une stratégie pour calculer les degrés de libertés*

Nous avons eu plusieurs choix pour entamer cette partie, voici quelques propositions qu'on s'est proposé avec les points forts et les points de limitation qui nous ont amené à choisir une stratégie et non pas une autre.

- **Première proposition (parcourir le goban) : *non efficace***
  la première proposition qu'on a eu étais parcourir le goban et si une intersection contient une pierre, on va attribuer à cette intersection une degré de liberté dans un autre tableau qui contient les degrés de libertés.
  Limitation : cette stratégie s'est avéré très limité, en effet une grande limitation de cette stratégie réside dans le cas où une intersection appartient à un groupe de pierres, voici une simulation de la limitation de cette stratégie

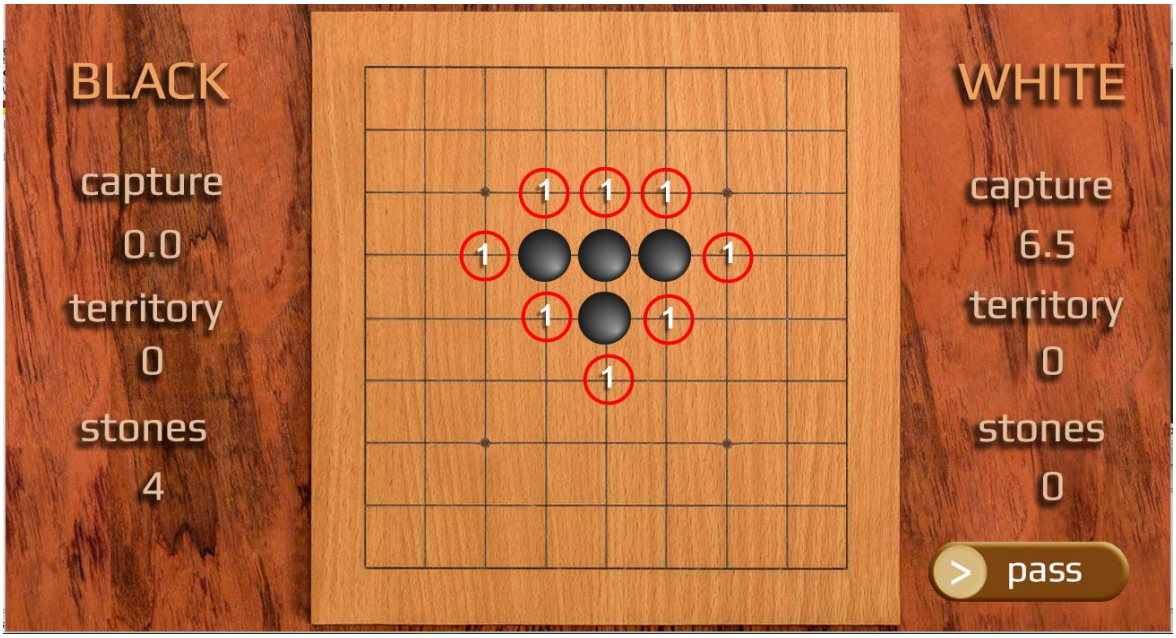


En effet, le problème qu'on aura par la suite si on a opté pour cette stratégie, c'est que un groupe de pierres peut être en Atari (degré de liberté=1) alors que la fonction lui attribuera une degré de liberté =2 parce que éventuellement une degré de liberté peut appartenir à deux pierres de la même couleur et par la suite sera calculé deux fois. Ceci va nous générer un problème au niveau de l'intelligence artificielle qu'on a voulu implémenter parce que dans ce cas notre AI va perdre une opportunité de capturé un groupe de pierres qu'elle aurai pu capturer facilement.

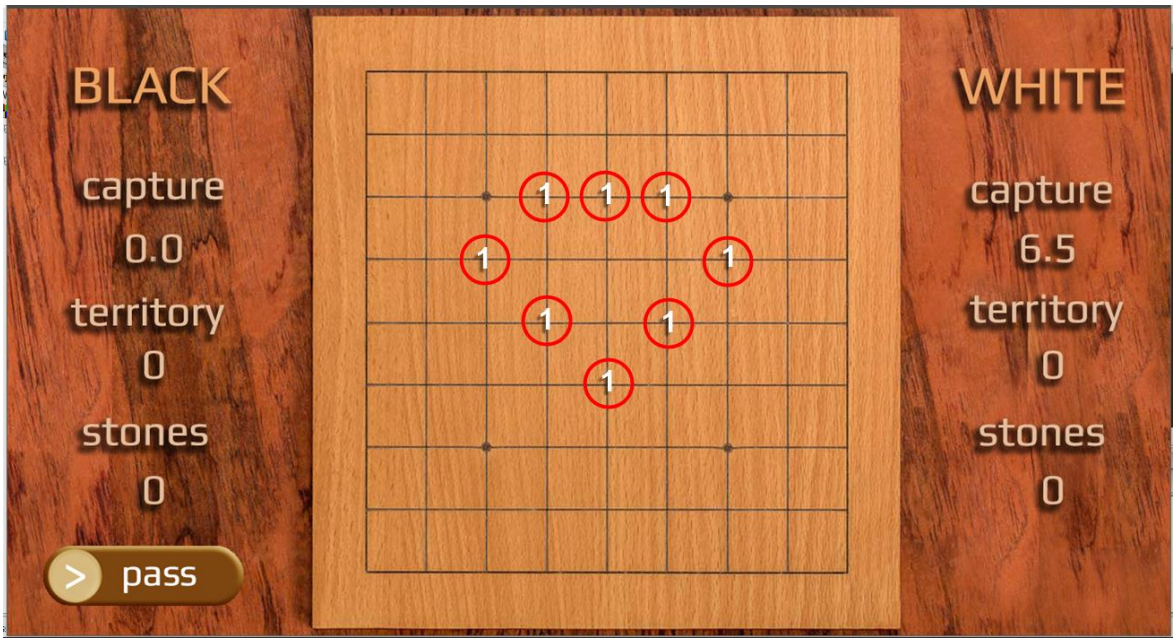
✓ **Conclusion** : cette stratégie de parcourir le goban et chercher à calculer les degrés de libertés n'est pas la stratégie la plus optimale.

- **Deuxième proposition (fonction récursive) : très efficace**
  La récursivité apporte souvent des solutions très optimales à des problématiques très compliqués, c'est pour cette raison qu'on a pensé d'utiliser une fonction récursive pour résoudre notre situation,

Voici comment on a procédé :
On prend par exemple cette situation, la même que précédemment.



La fonction récursive parcoure récursivement un groupe de pierres, et met dans un tableau les_degre_de_leberte[ ] le nombre 1 dans chaque intersection considéré comme degré de liberté de chaque pierre appartenant à ce groupe de pierres. Chaque fois qu'on parcoure une pierre on la met dans un tableau qu'on considère une blackliste pour ne pas parcourir une seule pierre 2 fois et puis tomber dans une boucle infinie. on obtient un tableau qui contient les éléments suivants :



Dans l'étape qui suit, on parcoure le tableau les_degre_de_leberte[ ] et on calcule combient de fois on tombe sur un nombre égale à 1 dans ce tableau, ce nombre c'est le degré de liberté du groupe de pierre qu'on a parcouru, et finalement on prend ce nombre qui est égale au degré de liberté de ce groupe de pierre et on l'attribue à chaque pierre de ce groupe de pierres.
Condition d'arrête : notre fonction récursive va s'arrêter lorsqu'elle aura parcouru toutes les pierres d'un groupe de pierres, et puis elle passera à un autre groupe et fera la même chose.

Maintenant qu'on a calculé toutes les degrés de libertés, il nous reste de capturer ce qu'il doit être capturer, c'est le rôle de la fonction capture( ), qui va parcourir le goban et éliminer toutes les pierres dont les degrés de libertés sont nulles.

Voici la fonction qui s'occupera de la capture, cette fonction prend une liste des pierres qui doivent être capturés qu'on a préparé avec la stratégie qu'on a discuter précédemment, et s'occupe de la capture en elle-même.

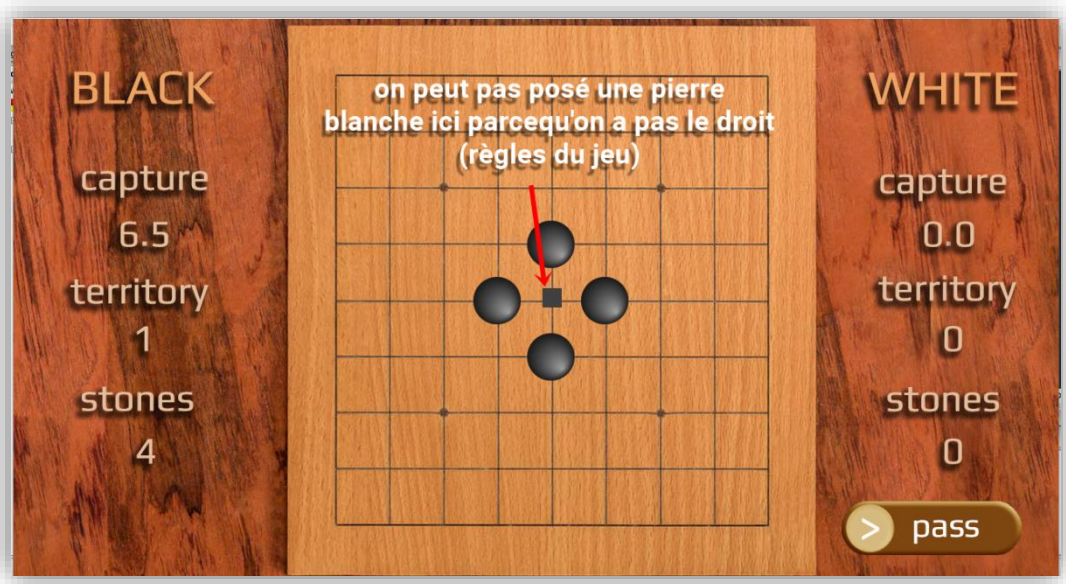
```
int capture()
{
    int to_return=0;
    for(int k=0;k<81;k++)
    {
        if(check2[k]==2)
        {
            if(data[k]==2)
            {
                player1_score=player1_score+1;
                data[k]=0;
                check2[k]=1;
            }
            if(data[k]==1)
            {
                player2_score=player2_score+1;
                data[k]=0;
                check2[k]=1;
            }
            to_return=1;
        }
    }
```

les pierres qui doivent etre capturés

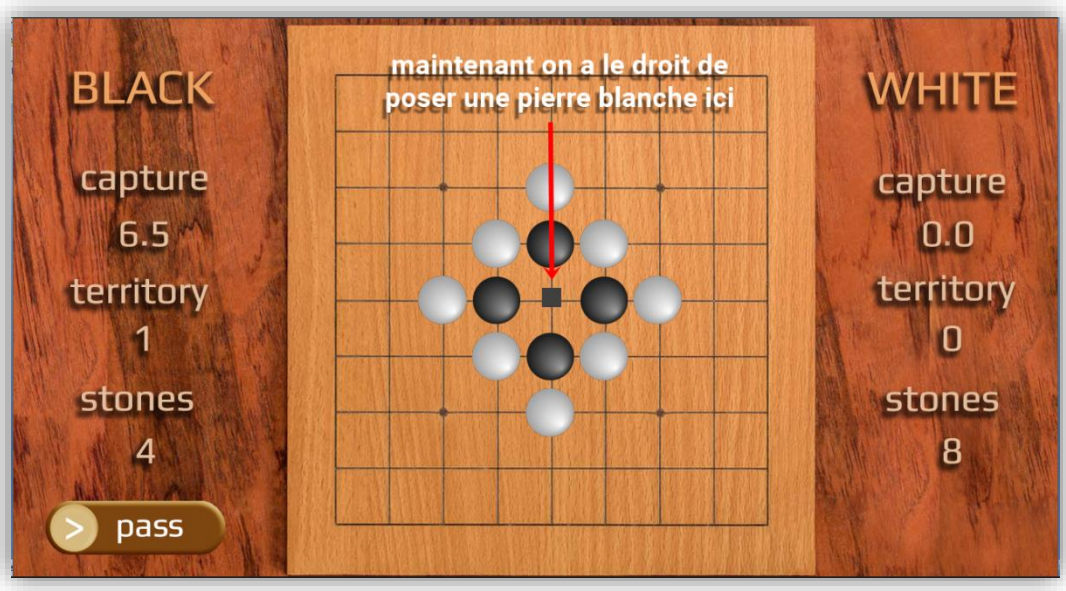
Après chaque capture on doit incrémenter le score de celui qui a capturé

9- Traiter un cas particulier du capture.

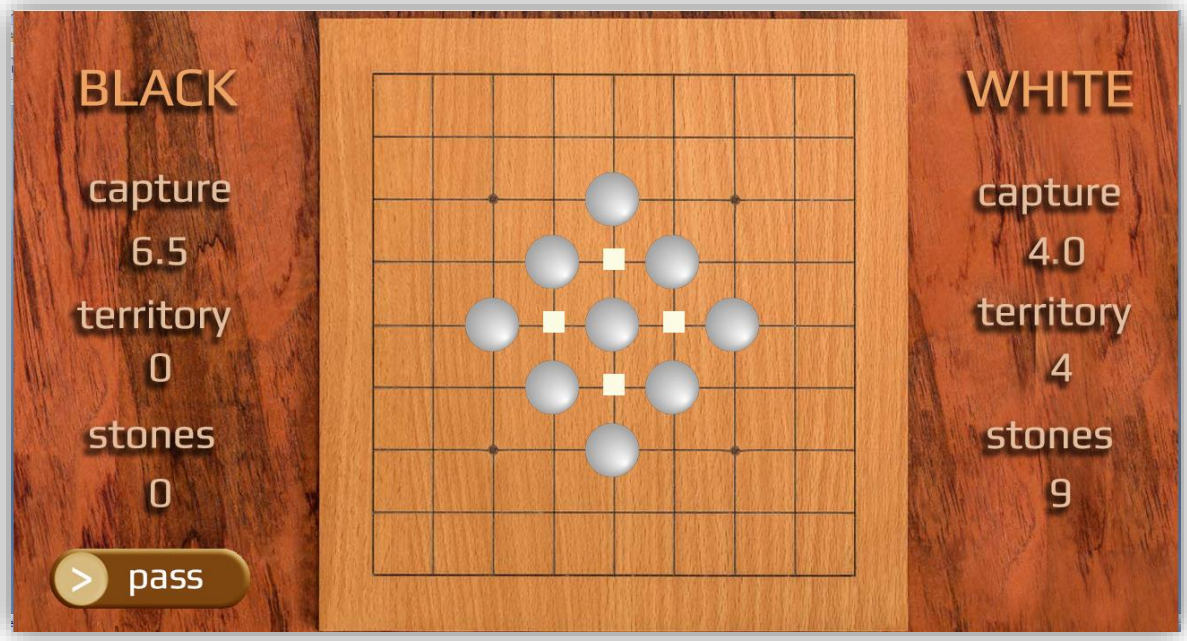
Un cas particulier qui a besoin d'un traitement propre et particulier :
Voici quelques figures illustrant le cas particulier qu'on va discuter qu'on a traité.



Effectivement, une pierre n'a le droit d'être posé sur une intersection de degré de liberté nulle que si elle pourra capturer (c'est une règle).
Maintenant voyant ce cas :



Effectivement lorsqu'on va poser une pierre sur l'intersection qu'on vient de motionner elle va capturer, c'est ce qui va se passer.



On a traité ce cas de figurer qui consiste en (une pierre ne pourra pas être posé sur une intersection si elle aura une degré de liberté nulle que si elle va capturer)

- ✓ **Remarque :** la simulation qu'on vient de montrer de ce cas, n'est qu'on cas particulier, bien sur nous on a implémenté un code qui va traiter tous les cas qui ressemble à cette situation.

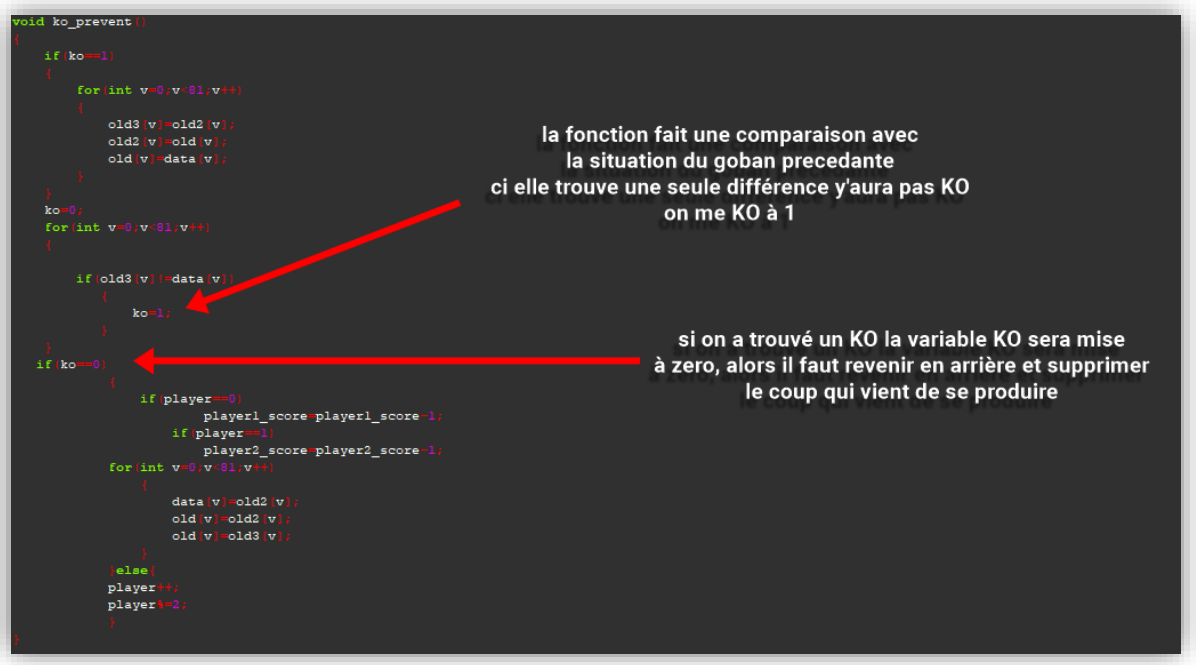
10- Traiter un cas particulier du jeu GO – le coup KO :

On a déjà expliqué le coup interdit KO dans la partie navigation dans le projet, maintenant on va juste expliqué comment on a traité ce cas.
On s'est proposé plusieurs propositions

- **Solution 1 : non efficace**
  La première solution qu'on s'est proposé étais de traiter chaque cas du KO indépendamment et la cette solution s'est avéré inefficace parce que il y'a beaucoup de cas de figures et en plus, il y avoir un cas de figure qu'on ne connaît pas.

- **Solution 2 : très efficace**
  La solution qu'on a implémenté c'est de stocker l'état du goban durant chaque tour de rôle, et plus comparer avec l'état du goban qui précède si elles sont les mêmes, on va revenir en arrière et supprimer le coup qui s'est produit.

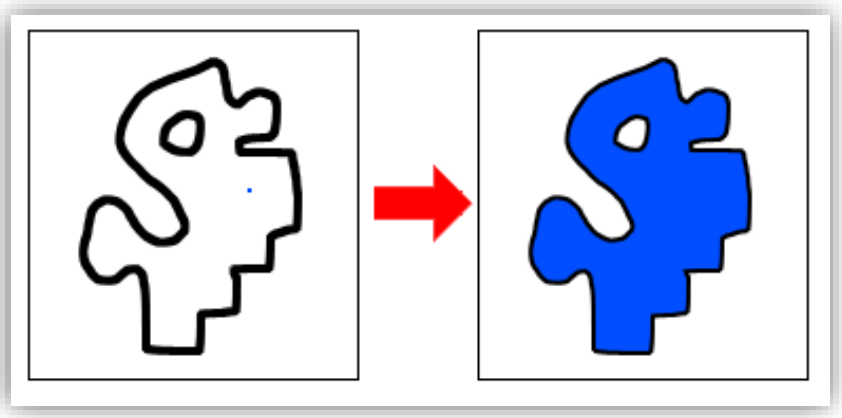
C'est pour cela qu'on a implémenter une fonction qui s'appel ko_prevent() cette fonction fait le traitement suivant, voici une aperçu du code



Un point fort de cette stratégie, c'est qu'on peut toujours abuser de son traitement pour implémenter un bouton qui permet de retourner en arrière une étape, parce que la le coup précédent de chaque coup est déjà stocker. Malheureusement on n'a pas eu de temps pour réaliser cette option mais elle est ouverte pour une éventuelle évolution.

11- Traitement des territoires :

Les territoires est par excellence la partie qui a pris le plus de temps dans la phase métrise du jeu.
On a déjà expliqué comment fonctionne le concept des territoires dans la partie concernant navigation dans le jeu. Dans cette partie on va aborder juste comment on a implémenté le concept des territoires.
On va commencer avec le tout début, on a posté sur cette problématique sur REDDIT, là on a trouvé l'idée qu'on a implémenté.
On s'est inespéré d'un algorithme qui s'appel FLOOD FILL algorithme, voici comment fonctionne l'algorithme du food fill.



Le concept est simple, c'est juste le mettre en œuvre qui est un peut compliqué. Le flood fill algorithme commence par un point et il se propage jusqu'à remplir une surface délimité par une bordure.

on a implémenter ce concept sous forme d'une fonction qui s'appel calcul_territoire( ) , fallait aussi apporter une codification, vue que un territoire ne doit pas être considéré comme territoire dès le moment où une pierre du joueur adverse est trouvé dans ce territoire.

int calcul_territoire(int i,int player_territory)

la fonction calcul_territoire qui fait cette tache, prend comme argument l'indice d'une intersection par laquelle elle commence pour constituer le territoire ainsi que le joueur qu'on veut lui calculer son territoire, et retourne le territoire auquel appartient le point i, on répète cette procédure pour tous les points vides du goban sauf les points qui sont déjà appartenant à un territoire qu'on a déjà constituer.

Cette fonction est une fonction récursive, elle s'appelle elle-même jusqu'à constituer un territoire qui est entièrement remplie ou bien jusqu'à trouver une pierre du joueur adverse, c'est la condition d'arrêt.

12- Affichage des scores :

Voila un autre aspect dans le jeu qui nous a posé problème dans la version SDL du jeu, effectivement c'est nécessaire d'afficher les scores des joueurs durant chaque tour de rôle, on veut dire par les scores, les captures, les territoires et les pierres sur le goban, c'est d'ailleurs les composantes qui entrent dans le calcul du score dans la version française du jeu go.

Le problème qu'on a eu, c'est dans l'affichage du score sur SDL, nous avons eu 2 solutions les voici avec les raisons qui nous ont poussés de choisir une et non pas une autre.

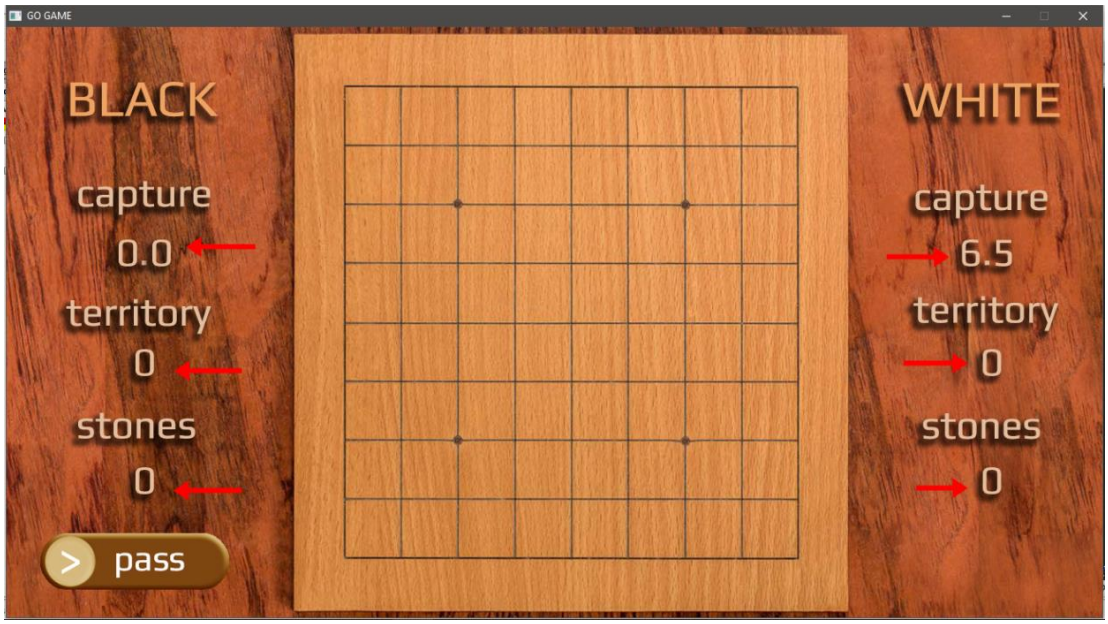
- **Solution 1 : éliminé**

Utiliser la bibliothèque SDL-ttf, c'est la solution que n'importe qui pourra se proposer dès le premier coup, le problème que posais cette solution, c'est que les nombres lorsqu'ils serons affichées il aurons pas le même design que l'arrière plan, je veut dire par le design, que les autres écritures qui existent déjà sur le background ont une ombre (shadow) et une police spécifique, la bibliothèque n'offre pas toutes ces possibilité du coup, les nombres lorsqu'ils serons afficher font avoir un allure qui n'est pas très élégant.

- **Solution 2 : meilleure solution**

La solution que nous avons trouvé et que nous avons estimé qu'elle sera la meilleur est la suivante.
Nous avons produits des images png pour les nombres entiers de 0 à 9, et nous avons implémenté une fonction qui fait des tests sur le score et puis elle affiche chaque entier séparément. Par exemple pour le nombre 123, la fonction va prendre l'image qui contient le nombre 1 et elle va l'afficher, et puis elle passera à 2 pour afficher l'image que contient le nombre 2 et de même pour le nombre 3. Et même pour la virgule si il y'en avait. Voici les images qu'on a utilisées pour effectuer cette opération, on les ai produits de tels façon qu'ils convient avec le design qu'on a utilisé.

Voici le résultat qu'on a obtenue par la méthode qu'on a fait :



On voit bien que les nombres imprimés ont une apparence meilleure et élégante.

13- Partie terminé et détermination du gagnant:

D'abord le gagnant sera déterminé par la méthode des règles française, le calcul du score total sera compté par la somme des captures, des territoires et des pierres que chaque joueur a sur le goban.
La partie termine si les deux joueurs ont passé, on a implémenté ce principe par une variable qui s'incrémente chaque fois qu'un joueur a passé, et qui est remise à zero chaque fois qu'un joueur fasse un coup sur le goban, quand cette variable prend la valeur 2 la partie est terminé.

14- Le joueur qui a le tour de rôle :

Pour modéliser le choix du joueur qui a le tour de rôle on s'est proposé deux méthodes :

- ✓ **Première méthode** : cette méthode consiste à mettre en place une variable entière qui prendra deux valeurs 1 et -1, et pendant chaque tour de rôle on va multiplier la variable par -1 pour passer la main au joueur suivant.

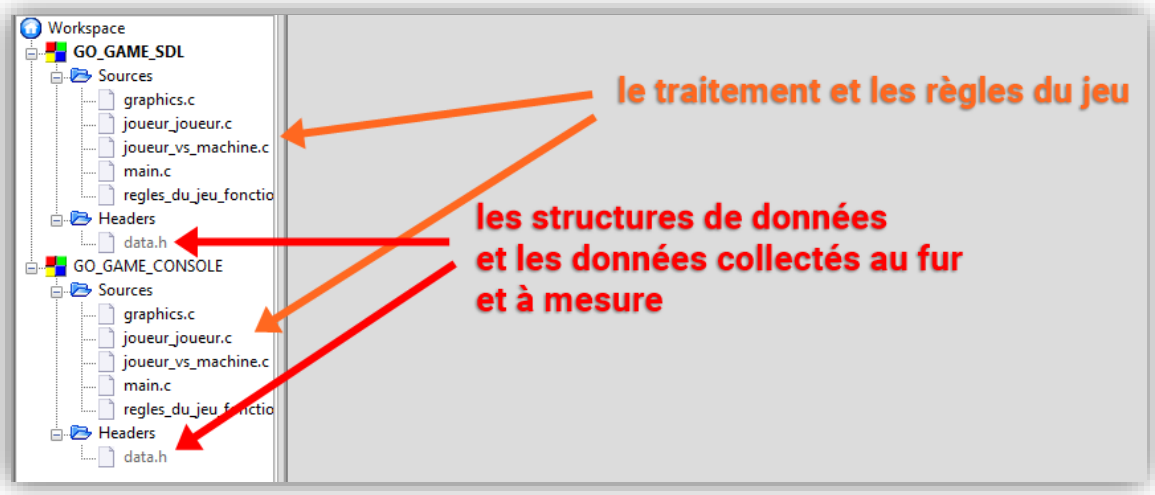
- ✓ **Deuxième méthode** : mettre en place une variable qui prendra les valeurs 0 et 1, on va switcher entre les deux valeurs pendant chaque tour de rôle en ajoutant 1 et en affectant au reste de la division sur 2.

- ✓ **Conclusion** : on a opté pour la deuxième méthode même si elle est plus lourde que la première méthode mais elle nous permettra de d'avoir la même convention avec player1 et player2 qu'on a déjà utilisé pour le stockage d'autres informations.

- ✓ **Remarque :** par la suite on a compris que cette décision n'étais pas très sage parce que l'opérateur arithmétique reste de la devisions euclidienne est une opération très lourde en terme de ressources.

VI. Architecture du jeu

Nous avons divisé le code source du jeu et on l'a distribué sur des fichiers .c et des fichiers .h pour le rendre plus lisible et plus accessible. Et pour alléger la fonction principale main qui doit représenter le bord de notre code, ce qui fait qu'elle ne doit pas être trop chargée.



- ✓ **Graphics.c :** contient les fonctions qui s'occupent de tous ce qui est interface IHM et affichage

- ✓ **Joueur_joueur.c :** contient tous les fonctions qui s'occupent de la partie entre deux humains

- ✓ **Joueur_vs_machine :** contient tous les fonctions qui s'occupent de la partie qui se déroule entre un humain et une machine.
- ✓ **Main.c :** c'est la fonction principale
- ✓ **Regles_du_jeu_fonctions :** ce fichier contient tous les fonctions qui s'occupent

VII. Intelligence artificiel

Malgré que l'intelligence artificielle soit une partie très importante dans le projet, malheureusement on n'a pas le temps suffisant pour lui attribuer tout le soin qu'elle mérite, vu tout le temps qu'on a pris pour travailler la partie territoires et les autres contraintes. Alors on a eu le temps pour implémenter juste deux stratégies pour d'intelligence artificiel.

1- Stratégie 1

La première stratégie qu'on a implémenté c'étais de donner une possibilité pour l'intelligence artificiel de capturer dès le moment où il y'a des conditions favorables pour la capture.
On a implémenté cette stratégie de telle façon que que lorsqu'une pierre du joueur adverse admet seulement 2 degré de liberté l'AI prendra directement la main pour aller capturer.

2- Stratégie 2

Pour la deuxième stratégie implémenté on a fait de telle sorte que lorsqu'une pierre du joueur de l'AI est en atari c'est-à-dire, elle admet seulement une degré de liberté, alors l'AI va prendre la main pour ajouter des pierres et augementer les degrés de libertés de sa pierre.

- ➢ **Remarque :** quand l'AI n'a pas de pierre à capturer ou bien de pierre ou un groupe de pierres de son couleur à faire survivre, dans ce cas la position pour jouer sera généré aléatoirement.
  L'AI qu'on a implémenté n'est pas complète malheureusement on n'a pas eu le temps de la travailler comme il faut, ce pour cela elle contient des failles.

VIII. Design du jeu

Le design est l'une des parties pour lesquelles on a attribué beaucoup de soin pour concevoir un produit qui plat à l'œil et qui est le plus élégant possible on a introduit tous les images constituant notre jeu dans le dossier img on a également inséré les fichiers photoshop PSD.

1- Le logo :

Voici le logo qu'on a conçu pour le jeu :

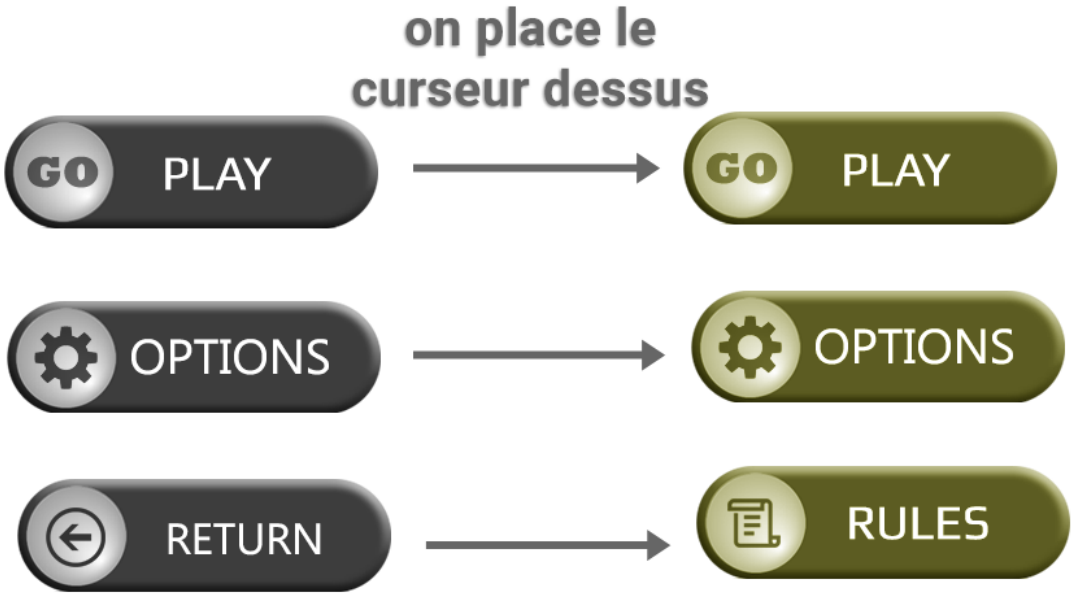
On a fait de telle sorte que le logo soit le convenable et expressif et élégant et qui convient l'image de l'arrière plan.

2- Image d'arrière plan :

Pour construire l'image de l'arrière plan on a pris une image qui contient des pierres du go, on a ajouté le logo et une sorte d'un brouillard pour que les boutons soient visible très bien, et puis nous avons ajouté une ambre pour les boutons.

- **Remarque :** nos avons figé l'ambre dans l'arrière plan et non pas avec les boutons parce que les boutons qu'on a implémenté changent de couleurs lorsqu'on place le curseur de la souris dessus, du coup à cause de cette méthode l'ambre lorsqu'elle s'applique plusieurs fois elle devienne très sombre et moins plaisante à l'œil.

3- Les boutons :



Les boutons qu'on a implémentés pour notre jeu sont des boutons dynamiques de telle façon que lorsqu'on place le curseur dessus ils changent de couleurs.

4- Autres fenêtres :

Voici d'autres fenêtres qu'on a utilisé pour montrer celui qui a le tour de role ou bien celui qui gagne.

➢ **Pour indiquer celui qui a le tour de rôle**

➢ **Pour indiquer celui qui gagne le paritie**

IX. Clôture :

Ce projet était une très bonne expérience pour moi, j'ai vraiment pris beaucoup de plaisir en travaillant ce jeu GO surtout parce que c'est le premier vrai projet que j'ai travaillé dans ma vie. J'ai pris ce travaille comme un challenge avec moi-même j'ai voulu me dépassé et produire quelque chose dont je serai fier, j'ai fait le maximum d'effort que j'ai pus fournir et je suis satisfait de mon travaille, même si j'ai voulu faire mieux (une version android) mais malheureusement le temps s'est écoulé.
Ce projet m'a permet d'apprendre beaucoup de choses en surmontant tout les difficultés que j'ai rencontré, d'ailleurs c'est le premier projet de programmation dans lequel j'ai utilisé une feuil et un stylo pour faire du vrai brain storming et du vrai conception d'un algorithme. Pour produire mon premier projet qui sort du cadre des exercices minimaux de l'algorithmique.