IFB 2020

Université Technologique de Belfort Montbéliard

Projet Informatique

Sujet : Réaliser une Belote Coinchée en C

Membres du projet :

- Julien CONSTANT
- Ewen BOURDON
- Théo SILVA

utbm
université de technologie
Belfort-Montbéliard

Introduction

Dans le cadre de nos études au sein de l'Université de Technologie de Belfort-Montbéliard. Nous avons eu pour but de réaliser, en tant que projet, une belote coinchée en C.

Ce projet commun ne fût pas des plus simples, dans la mesure où nous ne devions pas simplement programmer une belote classique mais avant tout une belote coinchée dans laquelle les règles diffèrent de la belote classique. Ce jeu permettra à un joueur seul de jouer avec trois IA en respectant les règles.

Afin de réaliser ce projet, nous avons décidé de créer un groupe de **trois** personnes dans la mesure où cela respectait la consigne d'un trinôme possible uniquement si l'effectif de la classe l'imposait. Dès lors notre groupe c'est composé **de** : **Ewen Bourdon**, **Julien Constant** & **Théo Silva**.

Afin de pouvoir travailler en groupe de manière convenable, il a été décidé de mettre en place un dépôt en ligne afin de permettre à tous d'accéder à la dernière version du projet, depuis chez-soi mais surtout depuis n'importe où. Nous avons donc choisi de prendre **GitHub**, notamment grâce à sa simplicité d'utilisation. En effet, ce dernier permet d'installer une application pour directement « **cloner** » le projet sur son ordinateur.

De façon globale, nous avons choisi de ne pas utiliser **Code::Blocks**, nous n'arrivions pas à comprendre sa façon de gérer les projets en C. De plus il restait très basique dans sa personnalisation. Ainsi nous nous sommes tournés vers **Sublime Text 3**, parfaitement modulable et très simple d'utilisation.

Table des matières

Introduction 2
1. Structure du projet 4
   A. Compilation : 4
   B. Les Headers 4
2. Analyse du projet 5
   A. Organigramme simplifié de la décomposition de la coinche 5
3. Solutions possibles et retenues 6
4. Description des algorithmes principaux 7
   A. Les Enchères : 8
      Explication détaillée : 8
   B. Les Plis : 9
Conclusion 10

1. Structure du projet

Le projet a été développé entièrement sous **Windows**, il n'est donc pas impossible de trouver des problèmes de compatibilités à l'exécution du programme sous certains **OS** tel que **Mac** ou **Linux**.

A. Compilation :

Afin de maintenir le projet aéré, il a été décidé de séparer les diverses fonctions dans des fichiers adjacents. Cependant, comme cité précédemment nous n'utilisions pas **Code::Blocks**, et **Sublime Text** ne permet pas la compilation de plusieurs fichiers en **.c**.

Nous nous sommes donc renseignés sur les différentes manières de compiler manuellement notre projet et avons choisi de compiler avec **gcc** puisqu'il était déjà installé sur nos machines. La création d'un « **makefile** » a donc été nécessaire pour renseigner les divers fichiers à compiler via **MingW**.

Grâce à ce makefile il nous suffit ensuite d'entrer simplement une unique commande dans le répertoire du projet pour lancer la compilation.

B. Les Headers

Les **headers** présents dans notre projet regroupent l'ensemble des fonctions dont nous avons besoins. Ainsi nous utilisons : **<stdio.h>** ; **<stdlib.h>** ; **<string.h>** ; **<time.h>** ; **<windows.h>** ; **<unistd.h>** et deux headers conçus par nos soins : <**fonctions.h**> & **<syntax.h>**

Le header **<fonctions.h>** possède l'ensemble des fonctions du programme, il est placé dans la majorité des fonctions et permet leur définition si elle n'a pas déjà été introduite. Il présente également un essai de compatibilité entre **Windows** et **Linux**. Cela reste un essai dans la mesure ou aucun d'entre nous ne possède **Linux** et n'a pu essayer si cela permettait la compatibilité à 100%. Cela implique donc la présence des headers **<windows.h>** & **<unistd.h>** dans ce dernier.

Le header **<syntax.h>** permet l'affichage de couleur, des entêtes et la mise en forme via l'attribution à des macros, permettant ainsi **d'augmenter la lisibilité** de notre programme et **d'uniformiser** notre code.

Les headers **<stdio.h>** ; **<stdlib.h>** & **<string.h>** sont présents car ils permettent l'utilisation des commandes usuelles du langage **C**, notamment via la gestion de chaîne de caractère, l'impression et la récupération de caractères sur la console.

Le header **<time.h>** permet simplement d'éviter l'indignation de notre compilateur lors de la compilation, en effet, la commande de l'aléatoire nécessite de façon indirecte cette bibliothèque.

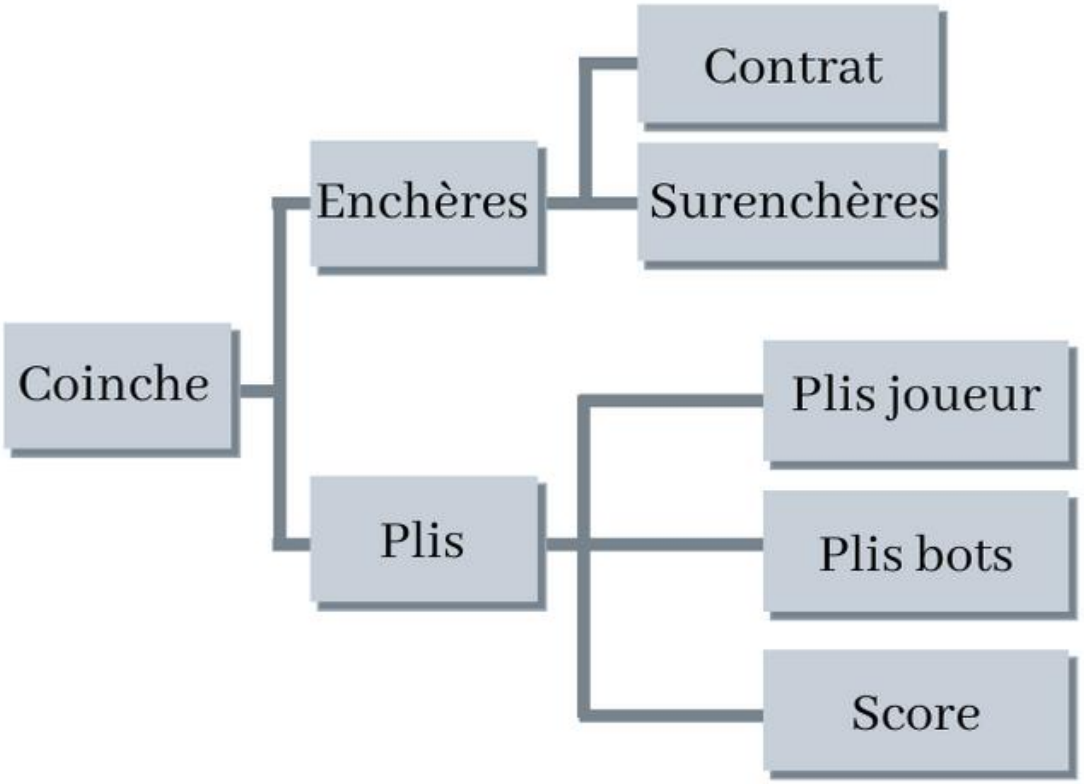
2. Analyse du projet

Afin de de pouvoir **maîtriser** le projet, il était nécessaire de se pencher sur **les règles** de la **coinche**, un jeu de carte auquel nous n'avions joués. Nous avons donc effectué un premier travail d'assimilation à ces règles pour faciliter la programmation de ce jeu.

Par la suite, nous nous sommes mis d'accord à la suite de quelques **échanges constructifs** des différents **obstacles** ou **zones d'incompréhension** auxquels nous devrions faire face lors de la programmation du projet.

Nous avons tout de suite compris qu'il serait **obligatoire** de séparer les différentes étapes du jeu et de différencier les programmes **affectant le joueur** et ceux qui permettront de **contrôler** les **IA** pour qu'elles respectent les règles du jeu. Ce découpage du jeu en diverses parties nous permet une plus grande lisibilité et une **efficacité accrue** lors de la programmation. Voici ci-dessous une présentation sommaire de la décomposition du jeu.

A. Organigramme simplifié de la décomposition de la coinche



3. Solutions possibles et retenues

Dans l'**analyse** du projet et des différentes **règles** nous avons trouvé plusieurs possibilités pour résoudre les différents **problèmes rencontrés** au cours de la programmation de notre projet. Prenons par exemple une des fonctions principales de notre **programme** qui consiste à faire jouer les **IA** pendant les plis. Nous avons pensé à deux solutions différentes :

- L'**IA** pose une carte de façon aléatoire (utile lors de la programmation de plis.c).

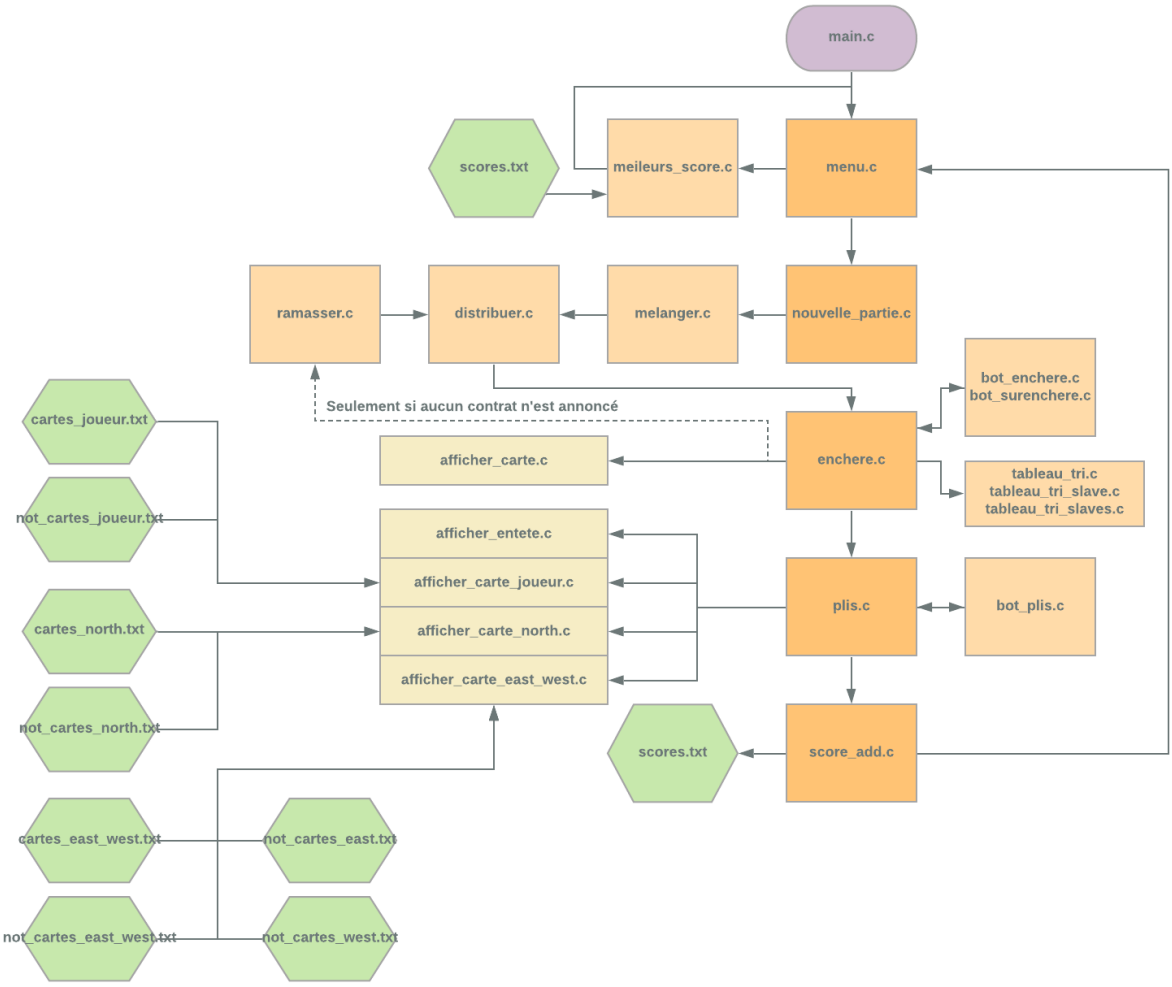
Utile dans un premier temps pour la création de la fonction, cette solution fût très vite délaissée car cela ne permettait pas aux **IA** de respecter les règles. Il fallait donc quelque chose plus proche des règles.

- L'**IA** joue selon le nombre de cartes jouées :
    - Si aucune carte n'a été posée, elle pose son atout le plus faible sinon elle pose sa carte la plus faible.
    - Si une ou plusieurs cartes ont été jouées, soit elle suit la couleur du pli en jouant sa carte la plus faible lui permettant de remporter le pli, soit elle joue son atout le plus faible lui permettant de remporter le pli.

Dans la mesure où nous devions faire **respecter** au maximum les règles pour les **IA**, nous avons choisi d'utiliser la deuxième solution, cette dernière permet le plus de se rapprocher d'une stratégie visant à gagner le plus efficacement. De plus cette solution convient mieux aux respects des règles spécifiques aux **IA** imposées dans le sujet. La seule contrainte de cette solution est le fait que les **IA** jouent tous leurs atouts dans les premiers plis et ne priorisent pas le reste de leur jeu.

4. Description des algorithmes principaux

Dans un premier temps, nous avons établi un **menu** permettant au joueur de choisir entre **commencer une partie**, **regarder les meilleurs scores** ou encore **quitter le programme**. Dès lors, nous avons créé au fil du besoin les diverses fonctions présente sur l'organigramme ci-dessous :

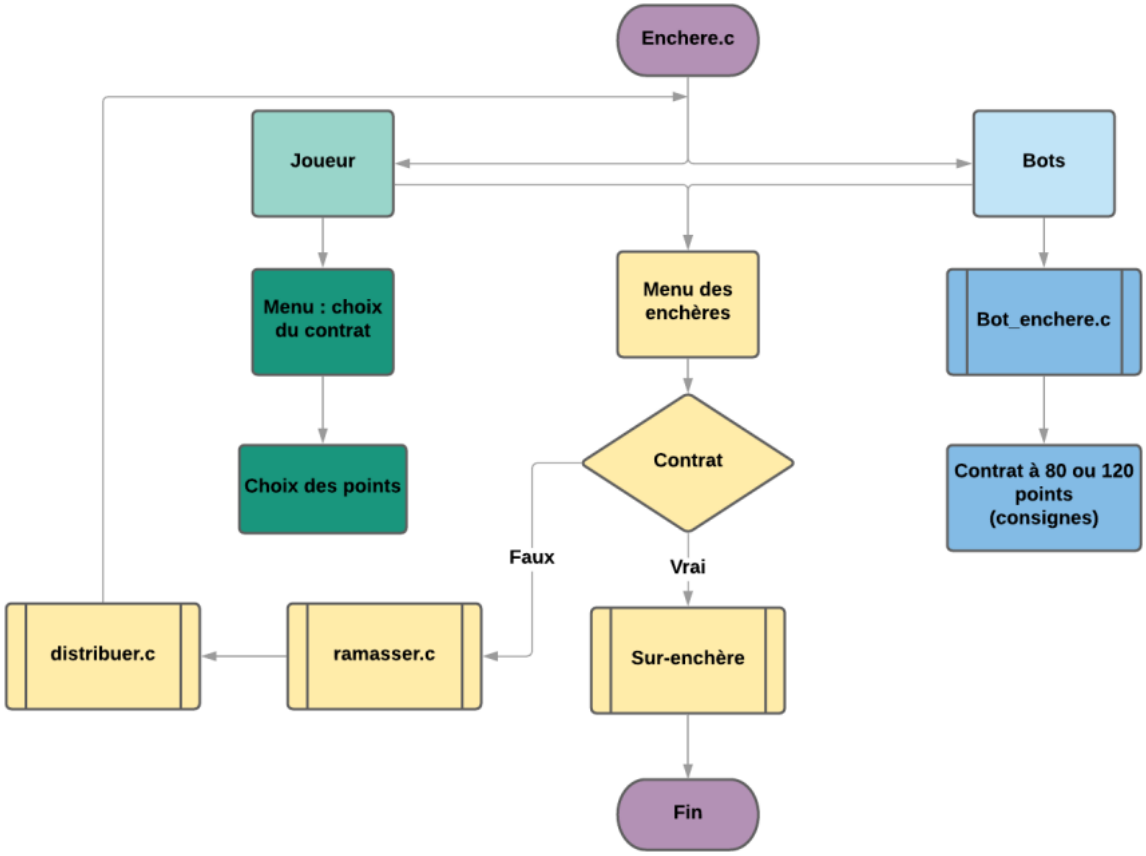


Parmi toutes les **fonctions** de notre projet, certaines sont plus importantes que d'autres comme les **plis** ou les **enchères**. Il est vrai que ces dernières représentent une grande partie de notre travail aux vues de leurs complexités et des résultats attendus. Nous allons **détailler** principalement ces deux fonctions afin de mieux **comprendre leurs fonctionnements**.

A. Les Enchères :

L'une des étapes **cruciales** de la Coinche est sa partie d'**enchère**, en effet c'est elle qui définie le déroulement de la partie. Durant les **enchères**, nous demandons tour à tour au joueur et aux IA d'**annoncer ou non** un **contrat** en fonction du jeu qu'ils possèdent. Si **aucun contrat** n'est annoncé, alors on **ramasse** les cartes dans l'ordre **sans** les **mélanger** (dans la mesure ou on ne mélange les cartes qu'au début de la partie, dans nouvelle_partie.c) et on les **redistribue** de la **même façon**, c'est-à-dire : 3 cartes par personnes, puis 2 et enfin 3.

Si **un contrat est annoncé**, on entre dans la phase des **surenchères**, si trois personnes passent d'affilés, alors on commence les plis.



Explication détaillée :

La première personne à annoncer et celle située à gauche du distributeur. Si aucun contrat n'est annoncé, la personne qui distribue se décale d'une place.

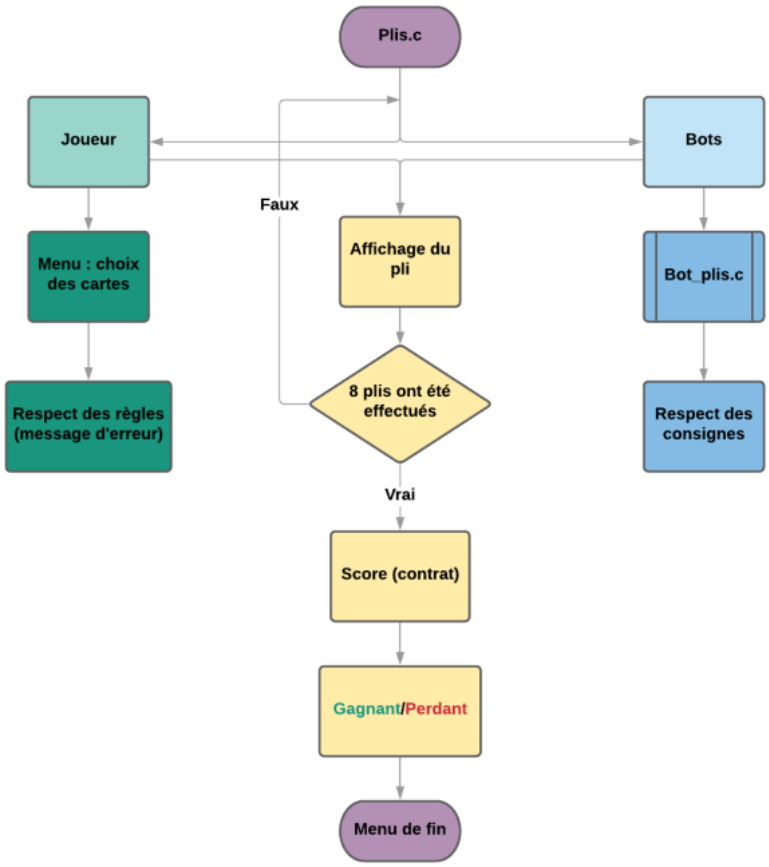
Les contrats sont annoncés par les IA dès qu'ils possèdent 3 cartes d'une couleur, il annonce alors 80 points, s'il en possède 4 alors ils annoncent 120 points. Si deux ou trois couleurs ont la même occurrence, l'IA décidera de manière aléatoire entre les couleurs.

Lors de la phase de surenchère, il est impossible pour les IA comme pour le joueur d'annoncer moins de points que l'annonce initiale.

B. Les Plis :

Les **plis** représentent les **tours de tables** et sont au nombre de **huit**. Ces derniers emmènent à ce que tous les joueurs est déposé leurs cartes, à la fin de chaque pli, on regarde qui remporte la main est on **attribue** les points en fonction de cela.

Là aussi comme pour les **enchères** on différencie le cas des **IA** et celui du joueur. C'est à travers cette fonction qu'on va **afficher** les différentes cartes que chaque joueur pose, **comptabiliser** les **points** à la fin du pli et par la suite renouveler un pli. Une fois les huit plis terminés, on affiche l'écran de fin, qui nous **informe** de notre **victoire** ou de notre **défaite**…



Explication détaillée :

Les plis se font selon une boucle se terminant lorsque 8 plis ont été effectués.

Le joueur possède un menu afin de voir ses cartes et celles du pli en cours. Nous avons programmé de manière à ce que le joueur soit obligé de respecter les règles. Les IA ont leur jeu qui est régi dans une autre fonction (expliquée dans le 3.).

A chaque fois que quelqu'un pose une carte, l'affichage est actualisé et les scores avancent à chaque pli. A la fin des 8 plis, on calcule le score des gagnants et on affiche les scores finaux ainsi qu'un visuel indiquant au joueur s'il a gagné ou perdu.

Conclusion

Durant ce **projet**, nous avons su **relever** les différents **problèmes** qui ont été mis en lumière lors de notre **analyse**. Un point clé à la réussite de ce projet a été la **communication** entre les membres de l'équipe. Ceci a été rendu possible grâce à **Discord,** qui nous permettait de nous **réunir** fréquemment et d'échanger diverses **informations** sur l'avancement du projet.

Le travail collectif était d'autant plus efficace grâce à l'utilisation de **GitHub,** qui nous a permis de suivre le travail des autres en temps réel et donc de **s'entraider** à la réalisation du **programme** et de mener à bien ce projet.

Ce projet nous a permis d'**approfondir** nos connaissances en cherchant sur différentes ressources, dont internet, des moyens ou des façons d'effectuer une tâche. De plus cette expérience nous a **appris** de nouvelle façon de percevoir et d'analyser un sujet précis ce qui nous sera utile pour de nombreuses situations tant bien professionnelles que personnelles. Ce projet nous a donner une vision plus précise des attentes qu'on a d'un **ingénieur en informatique**.