

Matthias MEUSBURGER
Groupe 1A

PROJET DE M.O.P.

Le jeu du cinquante et un



Sommaire :

1) Fonctionnalités de l'application :.....	3
2) Stratégies :.....	3
3) Statistiques :.....	4
4) Structure de l'application :.....	4
1) Remarques :.....	4
2) Diagramme des classes :.....	5
5) Structures de données :	6
6) Problèmes rencontrés durant le développement :	6
7) Remarques sur la programmation :.....	6
7) Jeu d'essai :.....	7
8) Annexe	10

1) Fonctionnalités de l'application :

- Déroulement d'une partie de 51.
- Choix du mode de jeu (ie : humain contre ordinateur ou ordinateur contre lui-même)
- Choix du niveau de jeu de l'ordinateur (faible ou fort)
- Affichage du résumé des parties jouées

2) Stratégies :

Le joueur incarné par l'ordinateur possède deux niveaux de jeu : faible et fort.

Dans le premier cas, la stratégie employée est plutôt basique. En effet, l'ordinateur se contente de jouer la plus grande carte qui ne fasse pas dépasser 51, et ce, pour contraindre l'adversaire à se retrouver très proche de la limite fatidique.

En revanche, le mode « fort » est plus subtil. En effet, l'ordinateur tient alors compte de sa main. Selon qu'il possède des cartes intéressantes ou non, il jouera de manière offensive ou défensive. Les cartes jugées intéressantes sont le 10 et le 9, puisqu'elles permettent de se tirer hors de danger lorsqu'on se trouve près de 51. Ainsi, à chaque tour, l'ordinateur pourra réévaluer son mode de jeu.

Celui-ci considérera qu'il peut jouer de manière offensive dès lors qu'il possède au moins deux cartes intéressantes. Le mode offensif repose sur le même principe que le niveau de jeu faible, à l'exception du dix, que l'on ne pose que si on a pas d'autre possibilité, en vue de le conserver, alors que le mode défensif repose sur le principe inverse : on joue la plus petite carte possible (hormis le neuf et le dix, puisque ce sont des cartes fortes), pour prolonger la partie et ainsi espérer récupérer des cartes intéressantes lors des tours suivants.

On peut souligner le fait que cette manière de procéder permet une sélection des cartes fortes, qui ne seront utilisées qu'en dernier ressort.

A noter également que le choix du niveau de jeu de l'ordinateur peut s'appliquer aux parties opposant un humain à l'ordinateur, mais aussi aux parties opposant l'ordinateur à lui-même. Ainsi, il est possible de faire jouer un ordinateur faible contre un ordinateur fort.

3) Statistiques :

Afin d'évaluer la performance des stratégies utilisées, j'ai procédé à des tests portant sur un grand nombre de parties.

Il en résulte que pour 100 parties, si l'on oppose un ordinateur faible à un autre faible, on obtient un ratio de 59/41.

Si l'on oppose un fort à un fort, cela donne un ratio de 48/52.

Et finalement, si l'on oppose un fort à un faible, on obtient un ratio de 84/16.

La dernière proportion tend à confirmer la justesse des stratégies développées.

4) Structure de l'application :

1) Remarques :

L'application est composée de neuf classes, que l'on peut diviser en trois catégories :

- celles qui ont trait au jeu
- celles qui ont trait aux joueurs
- et celles qui ont trait au jeu de carte

Une seule classe concerne le jeu, il s'agit de la classe éponyme.

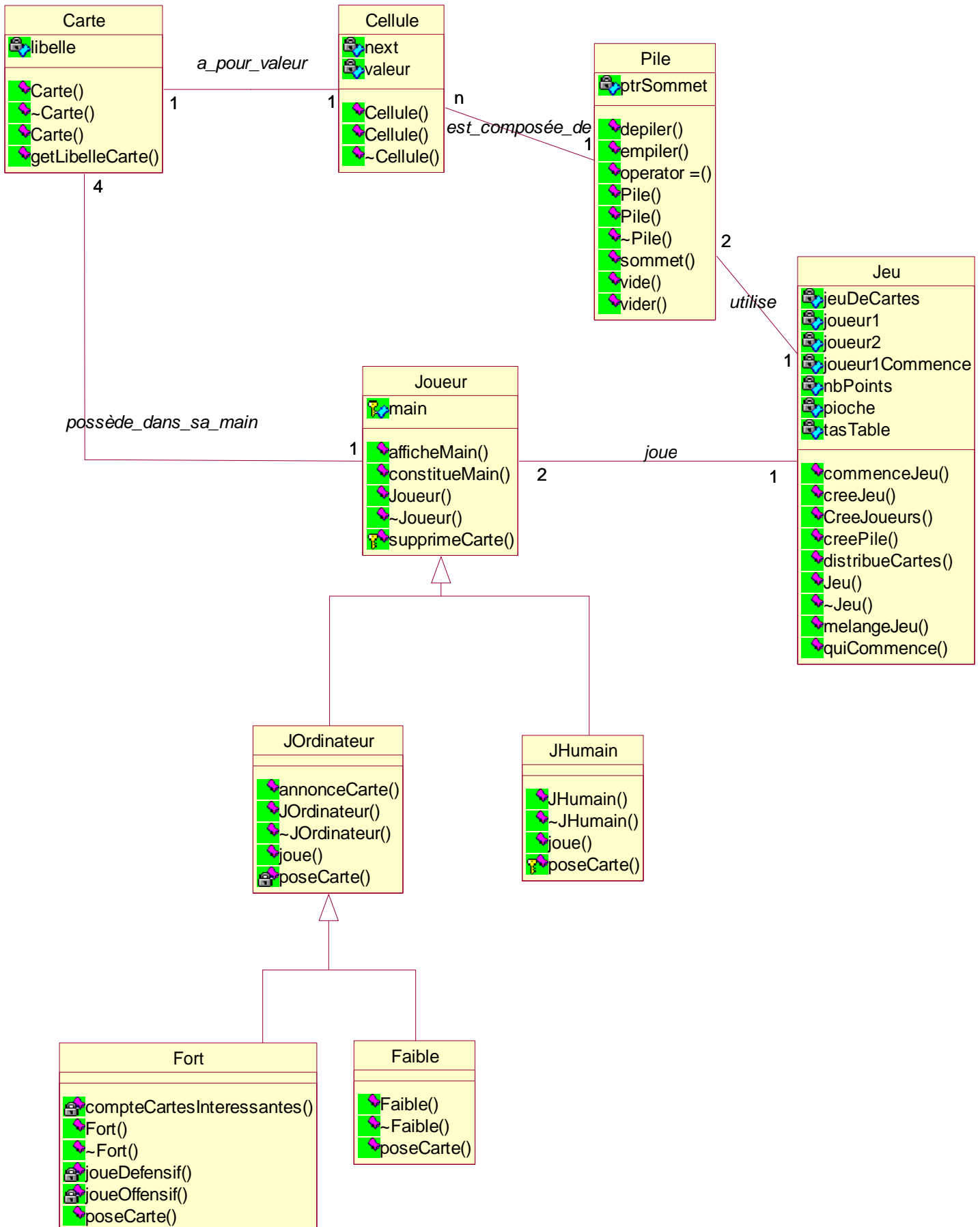
Quatre classes concernent les joueurs : Joueur, JHumain, JOrdinateur, Faible, et Fort.

Et trois classes concernent le jeu de carte : Carte, Cellule et Pile.

On peut remarquer la hiérarchie de classes concernant les joueurs : Faible et Fort sont filles de JOrdinateur, qui est fille de Joueur, tout comme JHumain. Cette organisation offre l'avantage non négligeable de pouvoir manipuler un joueur sans se soucier aucunement de savoir s'il est humain ou ordinateur, et s'il est fort ou faible. L'utilisation des objets joueurs se fait donc de manière totalement transparente pour les classes clientes.

En ce qui concerne la classe Pile, on pourra s'interroger sur le fait que celle-ci n'utilise pas un patron de classe, et donc qu'elle n'est pas réutilisable. Ce choix est motivé par un souci de clarté. En effet, la classe Pile ne comportant qu'un seul type de données dans toute l'application, les objets de type Carte, il m'a semblé non seulement superflu d'utiliser un patron de classe, mais de surcroît maladroit, puisque la lisibilité du code s'en serait trouvée gravement altérée.

2) Diagramme des classes :



5) Structures de données :

J'ai choisi d'utiliser une structure de données de type pile (LIFO¹) pour modéliser la pioche et le tas de cartes sur la table. Ce choix est évident, puisque piocher une carte revient à prendre celle qui est au sommet du tas, et poser une carte sur la table revient à l'empiler sur celles déjà présentes.

En ce qui concerne le jeu de cartes, j'ai choisi d'utiliser un tableau d'objets de type Carte, ce qui ne constitue pas forcément la solution la plus élégante, mais très certainement la plus maniable et la plus pratique.

6) Problèmes rencontrés durant le développement :

Lors d'une partie impliquant un humain et l'ordinateur, le jeu est rythmé par les questions posées à celui-ci. En revanche, lors d'une partie opposant l'ordinateur à lui-même, l'affichage du déroulement du jeu est ininterrompu. C'est pourquoi j'ai voulu introduire l'attente d'appui d'une touche entre chaque tour pour passer au tour suivant, à l'aide des fonctions getch() et getchar(). Cependant, ces arrêts se sont révélés être très imprécis. En effet, le programme attendait l'appui d'une touche parfois même lorsqu'il était en train d'annoncer la main ! Après un examen attentif de l'exécution de mon programme grâce au debugger de Visual C++, je me suis rendu compte que les affichages utilisant les flux (provenant de iostream.h), comme les instructions cout, pouvaient être différées par rapport à l'exécution, c'est-à-dire intervenir dans la console après l'instruction correspondante, tandis que les getch() et getchar() (provenant de conio.h) étaient pris en compte immédiatement. Je n'ai pas pu résoudre le problème induit par cette désynchronisation entre les deux méthodes d'affichage et de saisie. Ainsi, lors d'une partie opposant l'ordinateur à lui-même, aucune pause n'est prévue avant la fin de la partie. Il faut donc prévoir d'augmenter la mémoire de la console pour pouvoir remonter et suivre la partie.

7) Remarques sur la programmation :

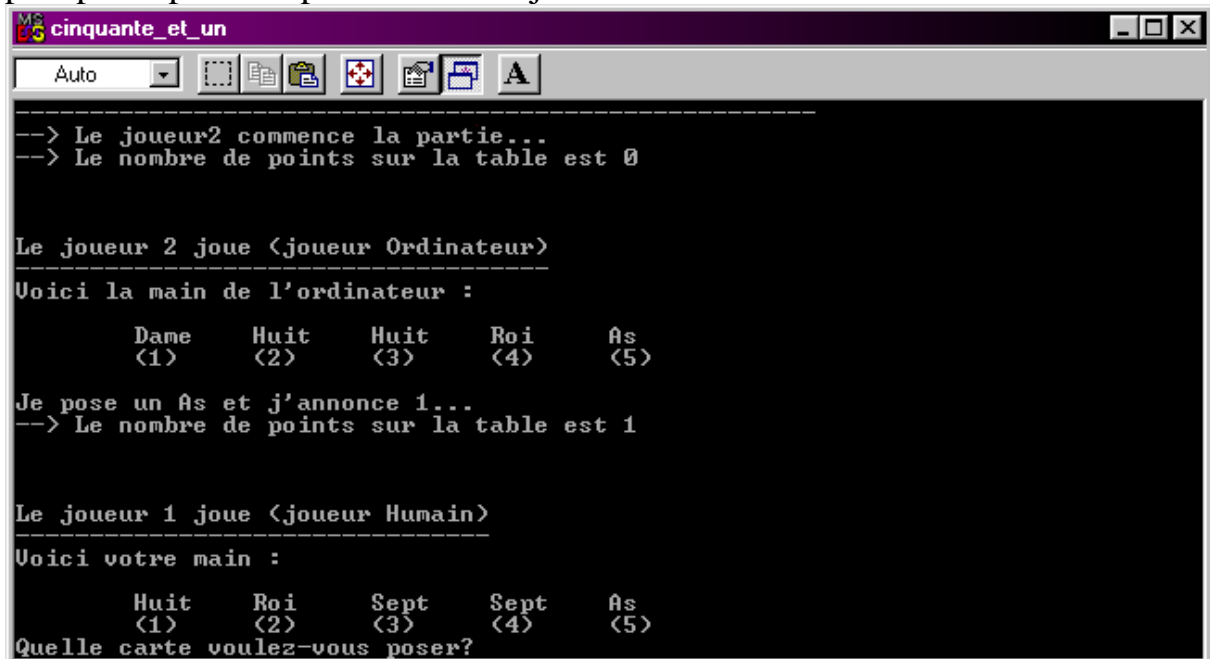
En ce qui concerne la structure du code, on peut noter l'utilisation de fonctions virtuelles au sein des classes Joueur et JOrdinateur, afin d'induire une ligature dynamique des fonctions concernées, et ce, toujours dans le but de garantir une utilisation transparente des objets de type joueur.

¹ LIFO : Last In First Out

7) Jeu d'essai :

Voici quelques exemples illustrant le comportement de l'ordinateur au cours d'une partie.²

En mode fort, si l'ordinateur n'a pas de cartes intéressantes, il joue la carte la plus petite possible pour ralentir le jeu.



```
cinquante_et_un
Auto
-----
--> Le joueur2 commence la partie...
--> Le nombre de points sur la table est 0

Le joueur 2 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :

      Dame      Huit      Huit      Roi      As
      (1)      (2)      (3)      (4)      (5)

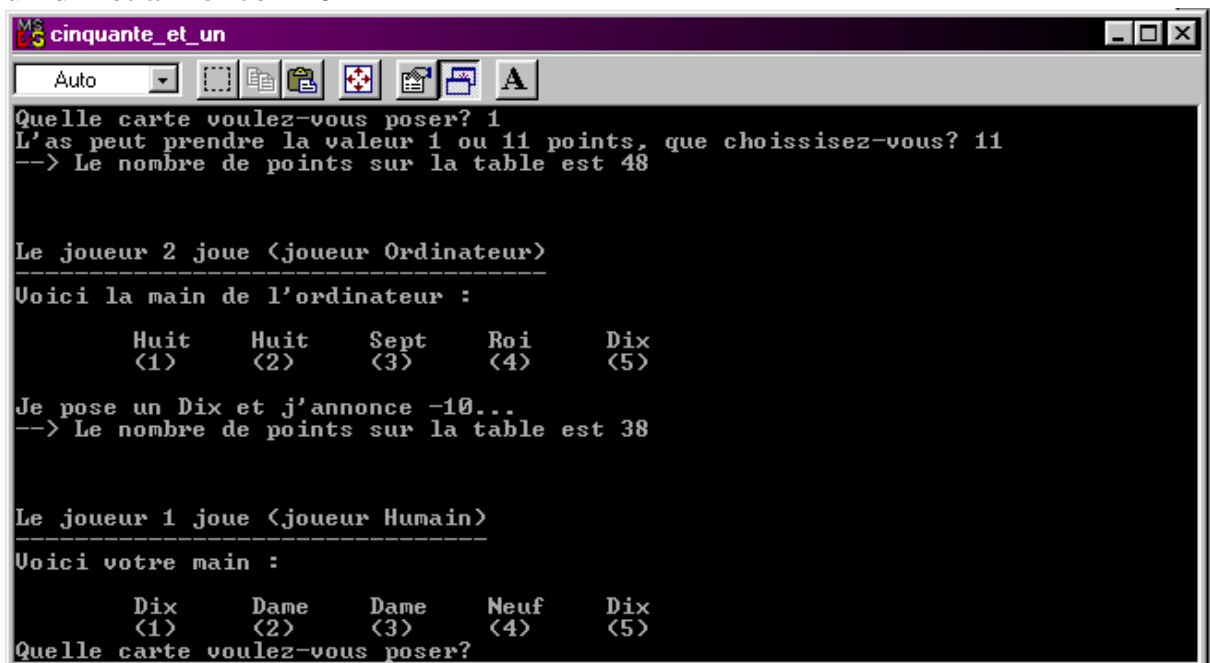
Je pose un As et j'annonce 1...
--> Le nombre de points sur la table est 1

Le joueur 1 joue <joueur Humain>
-----
Voici votre main :

      Huit      Roi      Sept      Sept      As
      (1)      (2)      (3)      (4)      (5)

Quelle carte voulez-vous poser?
```

L'ordinateur ne peut pas poser une carte « normale » sans perdre, il pose alors un dix et annonce -10



```
cinquante_et_un
Auto
Quelle carte voulez-vous poser? 1
L'as peut prendre la valeur 1 ou 11 points, que choisissez-vous? 11
--> Le nombre de points sur la table est 48

Le joueur 2 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :

      Huit      Huit      Sept      Roi      Dix
      (1)      (2)      (3)      (4)      (5)

Je pose un Dix et j'annonce -10...
--> Le nombre de points sur la table est 38

Le joueur 1 joue <joueur Humain>
-----
Voici votre main :

      Dix      Dame      Dame      Neuf      Dix
      (1)      (2)      (3)      (4)      (5)

Quelle carte voulez-vous poser?
```

² Ces exemples sont tirés d'une version spéciale du programme dans laquelle l'ordinateur affiche sa main avant de jouer, et ce, afin de se rendre compte de sa manière de réagir au cours du jeu. Il est évident que ce n'est pas le cas dans la version finale.

L'ordinateur est mode offensif car il a deux « bonnes cartes », il pose donc la carte la plus grande possible pour rapprocher au maximum l'adversaire de 51. Notez qu'il conserve le dix, qui est une bonne carte, même si il s'agit de la plus grande, car il est souhaitable de rester en mode offensif le plus longtemps possible.

```

cinquante_et_un
Auto
(1) (2) (3) (4) (5)
Quelle carte voulez-vous poser? 5
--> Le nombre de points sur la table est 8

Le joueur 2 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :
      Huit      Dix      Sept      Sept      Neuf
      (1)      (2)      (3)      (4)      (5)

Je pose un Huit...
--> Le nombre de points sur la table est 16

Le joueur 1 joue <joueur Humain>
-----
Voici votre main :
      Neuf      As      Dix      Neuf      Sept
      (1)      (2)      (3)      (4)      (5)
Quelle carte voulez-vous poser? _

```

En mode faible, bien qu'il ait de mauvaises cartes, l'ordinateur joue quand même sa carte la plus forte, et se rapproche donc de 51.

```

cinquante_et_un
Auto
(1) (2) (3) (4) (5)
Quelle carte voulez-vous poser? 1
--> Le nombre de points sur la table est 15

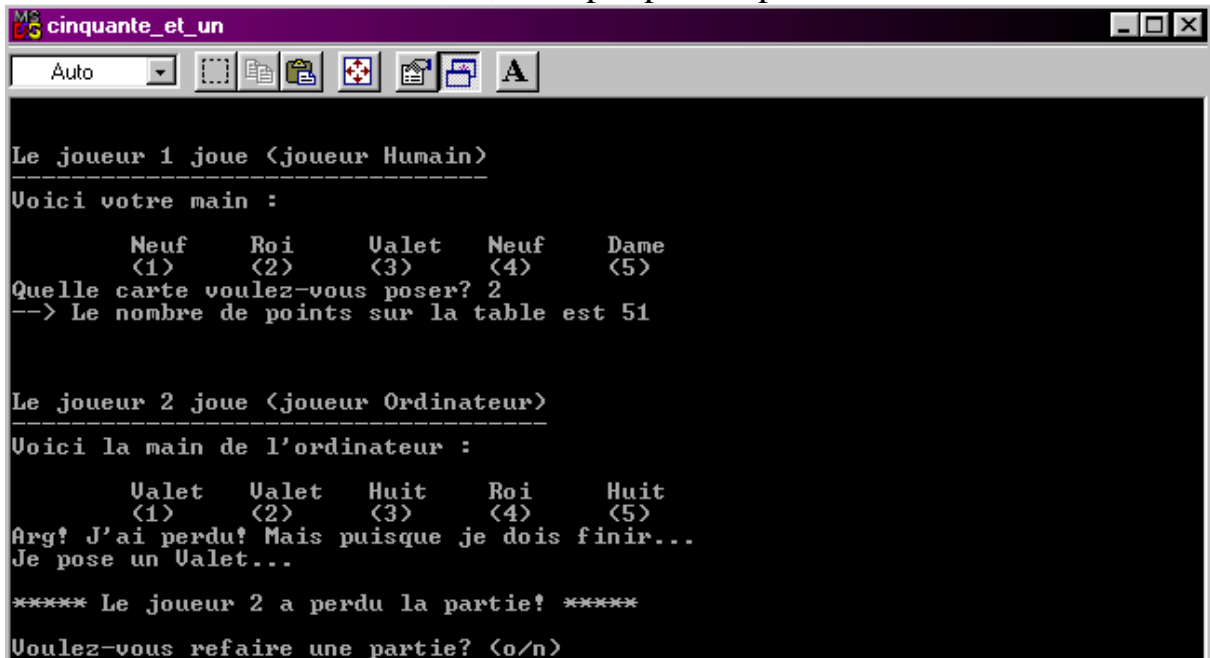
Le joueur 2 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :
      Sept      Huit      Roi      Neuf      Dame
      (1)      (2)      (3)      (4)      (5)

Je pose un Huit...
--> Le nombre de points sur la table est 23

Le joueur 1 joue <joueur Humain>
-----
Voici votre main :
      Neuf      As      Dix      Neuf      Roi
      (1)      (2)      (3)      (4)      (5)
Quelle carte voulez-vous poser? _

```

L'ordinateur avoue sa défaite, et, de dépit, pose la première carte de sa main.

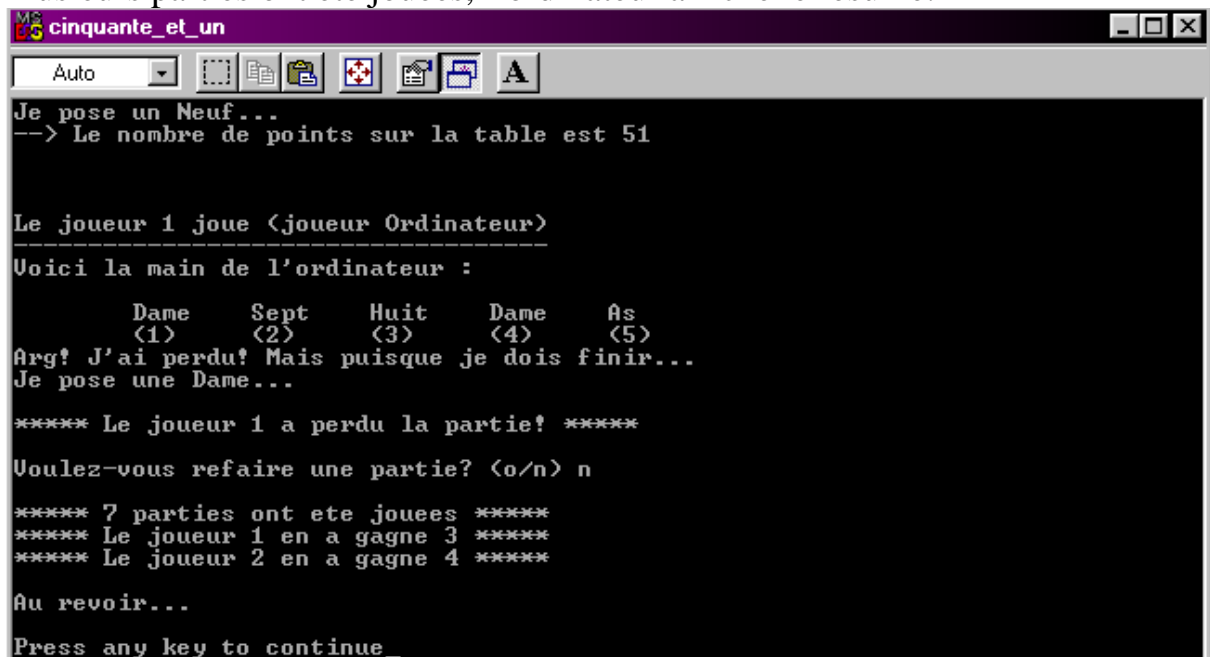


```
cinquante_et_un
Auto
Le joueur 1 joue <joueur Humain>
-----
Voici votre main :
      Neuf   Roi   Valet   Neuf   Dame
      (1)   (2)   (3)   (4)   (5)
Quelle carte voulez-vous poser? 2
--> Le nombre de points sur la table est 51

Le joueur 2 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :
      Valet  Valet  Huit   Roi   Huit
      (1)   (2)   (3)   (4)   (5)
Arg! J'ai perdu! Mais puisque je dois finir...
Je pose un Valet...

***** Le joueur 2 a perdu la partie! *****
Voulez-vous refaire une partie? <o/n>
```

Plusieurs parties ont été jouées, l'ordinateur affiche le résumé.



```
cinquante_et_un
Auto
Je pose un Neuf...
--> Le nombre de points sur la table est 51

Le joueur 1 joue <joueur Ordinateur>
-----
Voici la main de l'ordinateur :
      Dame   Sept   Huit   Dame   As
      (1)   (2)   (3)   (4)   (5)
Arg! J'ai perdu! Mais puisque je dois finir...
Je pose une Dame...

***** Le joueur 1 a perdu la partie! *****
Voulez-vous refaire une partie? <o/n> n

***** 7 parties ont ete jouees *****
***** Le joueur 1 en a gagne 3 *****
***** Le joueur 2 en a gagne 4 *****

Au revoir...
Press any key to continue_
```

8) Annexe : Codes sources