Programmation impérative Projet: Morpion 3D

Dinghao Li

December 2016

1 Contexte

Le but de ce projet est d'implanter une plateforme de jeu de type morpion 3D. Ce jeu se présente sous la forme d'un tableau de taille dans lequel sont placés des piles de jetons marquées par un X ou par un O.

$$taille = n \times n$$

Initialement aucun jeton n'est présent. Le jeu se joue à deux joueurs, l'un possède les jetons X, l'autre les jetons O. Le but du jeu est de faire apparaître une ligne de n jetons de son jetons.

A ttention ! Vous devez utiliser "gcc main.c menu.c keyboard.c player.c seisme.c -o jeu -lm" pour la compilation.

P our l'effectuer : "./jeu"

2 Introduction

Dans ce projet on realise 4 variantes total: Vue du dessus, 3D, Vue du dessus séisme et 3D séisme. On peut entrer les lettres pour effectuer les actions: choisir la variante, choisir la taille du plateau, choisir la hauteur maximum de la pile, déplacer le curseur de sélection de la case, poser un jeton (avec p), retirer un jeton (avec r), quitter le jeu directement (avec q). Si quelqu'un gagner selon la règle le jeu va retourner le message et se terminer.

3 Réalisation

3.1 Le Menu

Dans "menu.c" il y a les fonctions qui montrent l'interface principale et ceux qui représentent la variante différente. Chaque variante qui a le processus propre respectivement est une forme de la fonction soi-même par la modularisation.

Donc grâce à la modularisation dans la fonction "main.c" on peut seulement choisir la variante par utiliser les fonctions principales.

3.2 Les fonction de la chaque variante

U ne fonction représente une variante. Les variantes ont beaucoup des points communs qu'on peut réaliser ensemble et on pose les outil principals (les fonctions) dans le fichier "player.c". Vu les circonstances actuelles et on dois faire la simulation du "vrai" morpion 3D dans la vie réelle donc je choisis le tableau à trois dimensions (type char). Les avantages sont qu'on peut modifier la valeur dans le tableau facilement et pour la calculation plus vite. Alors dans chaque variante on peut choisir la taille n du plateau et la hauteur maximum h pour initialiser le plateau:

$$la\ taille\ de\ plateau = n \times n \times h$$

c 'est le code ci-dessous:

```
1
2  /*@require: Entier n pour la taille du plateau (n>0),et le plafond de la pile :h
3    @assigns: creer un espace pour plateau avec la taille n*n*n (type char)
4    @ensure : retour la pointeur du plateau (taille n) type char***
5  */
6  char*** init(int n,int haut)
7  {
8      int i,j;
9      char*** table=(char***)calloc(n,sizeof(char**));
10     for(i=0;i<n;i++)
11         table[i]=(char**)calloc(n,sizeof(char*));
12     for(i=0;i<n;i++)
13         for(j=0;j<n;j++)
14             table[i][j] =(char*)calloc(haut,sizeof(char));
15     return table;
16 }
```

L es réalisations du processus du jeu sont organisés par les fonctions.

3.3 Les joueurs

P our récupérer les déplacements du joueur plus automatique. Je choisis une fonction qui peut faire l'interaction avec le terminal directement.

```
1  static struct termios oldt;
2  /*/restore terminal settings*/
3
4
```

```c
/*@require: rien
  @assigns: rien
  @ensure : laisser le terminal normale
 */
void restore_terminal_settings(void)
{
  /*/Apply saved settings*/
    tcsetattr(0, TCSANOW, &oldt);
}


/*@require: rien
  @assigns: rien
  @ensure : lasser terminal obtenir un char dans un fois*/
void disable_terminal_return(void)
{
    struct termios newt;

    /*/save terminal settings*/
    tcgetattr(0, &oldt);
    /*/init new settings*/
    newt = oldt;
    /*/change settings*/
    newt.c_lflag &= ~(ICANON | ECHO);
    /*/apply settings*/
    tcsetattr(0, TCSANOW, &newt);

    /*/make sure settings will be restored when program ends*/
    atexit(restore_terminal_settings);
}
```

A vec ça quand les joueurs veulent déplacer le curseur c'est plus vite et pratique pour la réaction. Donc ce module est organisé dans le module "keyboard.c" et on va l'utiliser dans chaque situation pour déplacer le curseur (avec i,j,k,l). Puis dans chaque variante, il y a les fonctions pour les tours respectivement pour chaque joueur. Chaque tour,on récupère le choix du joueur et après on modifie le plateau pour effectuer les actions et on utilise une fonction qui vérifie qu'il gagne ou pas.

```c
/*@require: n>2 haut>0 plateau pas NULL
  @assigns: modifier le plateau selon le chiox de joueur
  @ensure : si X gagne on retourne 1 sinon retourne 0
 */
int Xtour(char*** plateau,int n,int haut)
{
    int *position;
```

```
8       char user[3];
9       char *control=user;
10      int res;
11      printf("\n   X    tour !!! \n ");
12      printf("\n         X tour:  p:ajouter    r:retirer   q:quitter  : ");
13      scanf("%c",control);
14      clearStdin();
15      while((*control!='p')&&(*control!='r')&&(*control!='q'))
16      {
17          printf("Taper 'p' ou 'r' ou 'q'\n");
18          scanf("%c",control);
19          clearStdin();
20      }
21      if(*control=='p')
22      {
23          position=choisirColonne(plateau,n,haut);
24          printf("%d,%d",position[0],position[1]);
25          res=ajouterX(plateau,haut,position[0],position[1]);
26          while(res==0)
27              {
28                  position=choisirColonne(plateau,n,haut);
29                  printf("%d,%d",position[0],position[1]);
30                  res=ajouterX(plateau,haut,position[0],position[1]);
31              }
32      }
33      else if(*control=='r')
34      {
35          if(PlateauEstVide(plateau,n,haut)==1)
36          {
37              printf(" \n le plateau est vide , tu peut seulment ajouter \n");
38              position=choisirColonne(plateau,n,haut);
39              printf("%d,%d",position[0],position[1]);
40              res=ajouterX(plateau,haut,position[0],position[1]);
41          }
42          else{
43                  position=choisirColonne(plateau,n,haut);
44                  printf("%d,%d",position[0],position[1]);
45                  res=tirer(plateau,haut,position[0],position[1]);
46                  while(res==0)
47                      {
48                          position=choisirColonne(plateau,n,haut);
49                          printf("%d,%d",position[0],position[1]);
50                          res=tirer(plateau,haut,position[0],position[1]);
51                      }
52              }
53      }
```

```
54          else if(*control=='q')
55              exit(1);
56          if(Xwin(plateau,n,haut,position[0],position[1])==1)
57          {
58
59              printf("\n !!!!! X WIN !!!!! \n");
60              return 1;
61          }
62          return 0;
63
64      }
```

3.4 Les choix

O n utilise un petit tableau pour conserver la position que les joueurs choisissent et puis on va transmettre la position à autre segmentation d'autres actions. Ici on utilise disable_terminal_return et restore_terminal_settings à changer la fonction du terminal (la fonction de récupérer les lettres automatiquement ou la situation normale).

```
1   /*@require: la pointeur de la plateau et la taille n>2,haut > 0
2     @assigns: creer pointeur: position[2] pour noter la position
3     @ensure : retour la position[2] que la utilisateur qui a choisi
4   */
5   int* choisirColonne(char*** plateau,int n,int haut)
6   {
7
8       int ch;
9       int* position = (int*)calloc(2,sizeof(int));
10      position[0]=0;
11      position[1]=0;
12      disable_terminal_return();
13      printf("choisir la pile a modifier \n");
14      /* Key reading loop */
15      while (1)
16       {
17          ch = getchar();
18          if (ch== 'i' )
19              {
20                  if(position[0]>0)
21                      position[0]=position[0]-1;
22              }
23          else if (ch== 'k' )
24              {
25                  if(position[0]<n-1)
26                      position[0]=position[0]+1;
```

```
27              }
28          else if (ch== 'j' )
29              {
30                  if(position[1]>0)
31                      position[1]=position[1]-1;
32              }
33          else if (ch== 'l' )
34              {
35                  if(position[1]<n-1)
36                      position[1]=position[1]+1;
37              }
38          /*/system("clear");*/
39          afficheChoisir(plateau,n,haut,position[0],position[1]);
40          if (ch == '\n')
41          {
42              printf("choirsirColone:␣%d,%d",position[0],position[1]);
43              restore_terminal_settings();
44              return position;  /* utiliser button space pour choisir*/
45
46          }
47       }
48
49      return 0;
50  }
```

3.5 Modification de plateau et affichage

I ci, on peut seulement modifier le plateau par modifier les valeur du tableau (char***). En même temps, on doit ajouter les limitations selon les cas différents (la pile est plein/vide...etc) et on doit mettre les restrictions et les rappels dans ces fonctions pour la stabilité du jeu selon la règle

```
1       int ajouterO(char*** plateau,int haut,int line,int colonne)
2       int ajouterX(char*** plateau,int haut,int line,int colonne)
3       int tirer(char*** plateau,int haut,int line,int colonne)
```

L'affichage est presque la part qui est un peu difficile. On doit montrer le tout plateau à les joueurs dans chaque tour. C'est-à-dire, il faut remonter le plateau selon la modification par les joueurs après chaque choix. On ajoute + à l'entour du curseur. Ici, on récupère la position que le joueur donne et on ajoute + en haut et en bas selon la ligne qu'il va déterminer qu'on va ajouter + avant la première ligne du jeton ou après la dernière (ou pas). Pour les deux cotés, on peut seulement imprimer le jeton entre deux + ('+%c+').

C 'est le code ci-dessous :

```
/*@require:pointeur p NOT NULL,n>2,haut>1,line >=0, colonne >=0
  @assigns:rien
  @ensure :affiche le pile que tu a choisi avec '+'
*/
void afficheChoisir(char*** p,int n,int haut,int line,int colonne)
{
    int x,y,z;
    int i,j;
    char show;
    int star=0;
    if(line==0)
    {
        for(i=0;i<n;i++)
        {
            if(i==colonne)
                printf("  +  ");
            else
                printf("      ");
        }
    }
    for(x=0;x<n;x++)
    {
        if((star==1))
        {
            printf("\n");
            for(j=0;j<n;j++)
            {
                if((j==colonne))
                {
                    printf("  +  ");
                }
                else
                    printf("      ");
            }
            star=0;
        }
        else
            printf("\n");
        if(x>0)
            printf("\n");

        for(y=0;y<n;y++)
        {
            show='.';
```

```
45                  for(z=0;z<haut;z++)
46                  {
47                      if(p[x][y][z]!='.')
48                          show=p[x][y][z];
49                  }
50                  if((x==line)&&(y==colonne))
51                  {
52                      printf("␣+%c+␣",show);
53                      star=1;
54                  }
55                  else
56                      printf("␣␣%c␣␣",show);
57                  if(x+1==line)
58                      star=1;
59              }
60          }
61          printf("\n");
62          if(line==n-1)
63          {
64              for(i=0;i<n;i++)
65              {
66                  if(i==colonne)
67                      printf("␣␣+␣␣");
68                  else
69                      printf("␣␣␣␣␣");
70              }
71          }
72          printf("\n");
73          printf("\t\tle␣pile␣basse->haut␣:␣");
74              for(i=haut-1;i>-1;i--)
75              {   printf("\n");
76                  printf("\t\t\t␣␣␣␣␣␣␣␣␣%c␣",p[line][colonne][i]);
77              }
78
79          printf("\n-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-\n\n\n");
80
81      }
```

3.6 Les conditions de la victoire

O n peut utiliser deux fonctions du jugement pour Vue de dessus et 3D parce que le séisme ne peut pas influencer les conditions de la victoire. L'idée de l'algorithme principal est qu'on a seulement de besoin de vérifier les conditions de la victoire à partir de la pile que les joueurs viennent de modifier. Ce n'est pas nécessaire qu'on vérifie tous les points dans le plateau. Dans la variante

'Vue de dessus', il est facile de vérifier quand on obtient tous les toits de pile.
Après on le vérifie sur ligne, colonne et deux diagonales.
Le code ci-dessous :

```
1     /******dessus*******/
2  /*@require: n>2 haut,line,colonne>0
3    @assigns: rien
4    @ensure : si X gagne on retourne 1 sinon retourne 0
5   */
6  int Xwin(char*** plateau,int n,int haut,int line,int colonne)
7  {
8      int i;
9      int nb=0;
10     /******line****/
11
12     for(i=0;i<n;i++)
13         {
14             if(pileToit(plateau,haut,line,i)=='X')
15                 nb=nb+1;
16
17         }
18     printf("␣line␣%d",nb);
19     if(nb==n)
20         return 1;
21     /******colone****/
22     nb=0;
23     for(i=0;i<n;i++)
24         {
25             if(pileToit(plateau,haut,i,colonne)=='X')
26                 nb=nb+1;
27         }
28     printf("␣colonne␣%d",nb);
29     if(nb==n)
30         return 1;
31     /****diag*****/
32     nb=0;
33     for(i=0;i<n;i++)
34         {
35             if(pileToit(plateau,haut,i,i)=='X')
36                 nb=nb+1;
37         }
38     printf("␣diag␣%d",nb);
39     if(nb==n)
40         return 1;
41     /*****diag2****/
42     nb=0;
```

```
43      for(i=0;i<n;i++)
44          {
45              if(pileToit(plateau,haut,n-1-i,i)=='X')
46                  nb=nb+1;
47          }
48      printf("\ndiag2\n%d",nb);
49      if(nb==n)
50          return 1;
51      return 0;
52  }
```

D ans la variante 3D, on vérifie les conditions à partir de la pile que les joueurs viennent de modifier aussi et dans ce cas, on a plusieurs situations.

Sur un même niveau :

On récupère le niveau du jeton que les joueurs viennent d'ajouter et on vérifie les sur cet niveau.

```
1   /*@require: n>1 haut>0
2     @assigns: rien
3     @ensure : si O gagne on retourne 1 sinon retourne 0
4    */
5    int Owin3D(char*** plateau,int n,int haut,int line,int colonne)
6   {
7       int i;
8       int nb=0;
9       int niveau=getNiveau(plateau,haut,line,colonne);
10      printf("\n\nniveau:%d\n\n",niveau);
11      /********sur la meme niveau*******/
12      /***line**/
13      for(i=0;i<n;i++)
14          if(plateau[line][i][niveau]=='O')
15              nb=nb+1;
16      if(nb==n)
17        return 1;
18
19      /***colonne***/
20      nb=0;
21      for(i=0;i<n;i++)
22          if(plateau[i][colonne][niveau]=='O')
23              nb=nb+1;
24      if(nb==n)
25          return 1;
26      /***diag****/
27      nb=0;
28      for(i=0;i<n;i++)
29          if(plateau[i][i][niveau]=='O')
```

```
30                  nb=nb+1;
31          if(nb==n)
32              return 1;
33          ///diag2///
34          nb=0;
35          for(i=0;i<n;i++)
36              if(plateau[n-1-i][i][niveau]=='O')
37                  nb=nb+1;
38          if(nb==n)
39              return 1;
```

P our 3D, on vérifie les lignes différentes en escalier sur une ligne, une colonne ou une diagonale. Ici, on regarde chaque jeton de cette ligne à partir de la pile que les joueurs viennent de modifier encore et tester le nombre de jetons sur la ligne dans le plateau. Si le nombre est suffisant, il va envoyer un message de la victoire.

```
1           ///////////////sur colonne(fixe) //////////
2           ///// montrer ////
3           nb=0;
4           int l=line;
5           int c=colonne;
6           int niv=niveau;
7           while((l<n)&&(niv<haut))
8           {
9               if(plateau[l][c][niv]=='O')
10                  nb=nb+1;
11              l=l+1;
12              niv=niv+1;
13          }
14          l=line-1;
15          c=colonne;
16          niv=niveau-1;
17          while((l>=0)&&(niv>=0))
18          {
19              if(plateau[l][c][niv]=='O')
20                  nb=nb+1;
21              l=l-1;
22              niv=niv-1;
23          }
24          if(nb==n)
25              return 1;
26          ///// descendre /////
27            nb=0;
28          l=line;
29          c=colonne;
```

```
30      niv=niveau;
31      while((l>=0)&&(niv<haut))
32      {
33          if(plateau[l][c][niv]=='O')
34              nb=nb+1;
35          l=l-1;
36          niv=niv+1;
37      }
38      l=line+1;
39      c=colonne;
40      niv=niveau-1;
41      while((l<n)&&(niv>=0))
42      {
43          if(plateau[l][c][niv]=='O')
44              nb=nb+1;
45          l=l+1;
46          niv=niv-1;
47      }
48      if(nb==n)
49          return 1;
50      /*////////////////sur line(fixe) //////////*/
51      /*/////////montrer////////*/
52      nb=0;
53      l=line;
54      c=colonne;
55      niv=niveau;
56      while((c<n)&&(niv<haut))
57      {
58          if(plateau[l][c][niv]=='O')
59              nb=nb+1;
60          c=c+1;
61          niv=niv+1;
62      }
63      l=line;
64      c=colonne-1;
65      niv=niveau-1;
66      while((c>=0)&&(niv>=0))
67      {
68          if(plateau[l][c][niv]=='O')
69              nb=nb+1;
70          c=c-1;
71          niv=niv-1;
72      }
73      if(nb==n)
74          return 1;
75       /*////////////////sur diagonale 3D(fixe) //////////*/
```

```
76      /*/////////1-montrer////////*/
77      nb=0;
78      l=line;
79      c=colonne;
80      niv=niveau;
81      while((c<n)&&(niv<haut)&&(l<n))
82      {
83          if(plateau[l][c][niv]=='O')
84              nb=nb+1;
85          c=c+1;
86          l=l+1;
87          niv=niv+1;
88      }
89      l=line-1;
90      c=colonne-1;
91      niv=niveau-1;
92      while((c>=0)&&(niv>=0)&&(l>=0))
93      {
94          if(plateau[l][c][niv]=='O')
95              nb=nb+1;
96          c=c-1;
97          l=l-1;
98          niv=niv-1;
99      }
100     printf("nb : %d",nb);
101     if(nb==n)
102         return 1;
103     /*/////1-descendre /////*/
104     nb=0;
105     l=line;
106     c=colonne;
107     niv=niveau;
108     while((c<n)&&(niv>=0)&&(l<n))
109     {
110         if(plateau[l][c][niv]=='O')
111             nb=nb+1;
112         c=c+1;
113         l=l+1;
114         niv=niv-1;
115     }
116     l=line-1;
117     c=colonne-1;
118     niv=niveau+1;
119     while((c>=0)&&(niv<haut)&&(l>=0))
120     {
121         if(plateau[l][c][niv]=='O')
```

```
122                 nb=nb+1;
123             c=c-1;
124             l=l-1;
125             niv=niv+1;
126         }
127         if(nb==n)
128             return 1;
129         /*/////////2-montrer////////*/
130         nb=0;
131         l=line;
132         c=colonne;
133         niv=niveau;
134         while((c>=0)&&(niv<haut)&&(l<n))
135         {
136             if(plateau[l][c][niv]=='O')
137                 nb=nb+1;
138             c=c-1;
139             l=l+1;
140             niv=niv+1;
141         }
142         l=line-1;
143         c=colonne+1;
144         niv=niveau-1;
145         while((c<n)&&(niv>=0)&&(l>=0))
146         {
147             if(plateau[l][c][niv]=='O')
148                 nb=nb+1;
149             c=c+1;
150             l=l-1;
151             niv=niv-1;
152         }
153         if(nb==n)
154             return 1;
155         /*/////2-descendre /////*/
156         nb=0;
157         l=line;
158         c=colonne;
159         niv=niveau;
160         while((c>=0)&&(niv>=0)&&(l<n))
161         {
162             if(plateau[l][c][niv]=='O')
163                 nb=nb+1;
164             c=c-1;
165             l=l+1;
166             niv=niv-1;
167         }
```

```
168         l=line -1;
169         c=colonne+1;
170         niv=niveau+1;
171         while((c<n)&&(niv<haut)&&(l>=0))
172         {
173             if(plateau[l][c][niv]=='O')
174                 nb=nb+1;
175             c=c+1;
176             l=l-1;
177             niv=niv+1;
178         }
179         if(nb==n)
180             return 1;
```

Finalement, c'est pour n jetons identiques consécutifs dans un même niveau. À partir de le jeton que les joueurs viennent d'ajouter, on va vérifier que s'il y a n jetons identiques consécutifs en dessous.

```
1           /*/////////////sur la pile/////////////////*/
2
3         nb=0;
4         l=line;
5         c=colonne;
6         niv=niveau;
7         while((niv>=0)&&(plateau[l][c][niv]=='O'))
8         {
9             if(plateau[c][l][niv]=='O')
10                nb=nb+1;
11            niv=niv-1;
12        }
13        if(nb==n)
14            return 1;
15        /*////personne gagne////*
16        return 0;
17    }
```

4 Les difficulté rencontrer et les solution

4.1 Scanf

Q uand j'utilise la fonction "scanf" avec %c je trouve qu'il y a les bugs avec les inputs par hasard. Il ne peut pas nettoyer le flux (la fonction fflush(stdin) ne fonctionne pas dans LIUNX) du input et distinguer le chiffre supérieur à dix. Donc j'utilise "string" pour les inputs et ajoute une fonction d'effacer le flux du input 'getchar()' et je choisi la fonction "strtol" au lieu de "atoi".

```
/*@require: rien
  @assigns: rien
  @ensure : effacer le flux du input(stdin)
*/
void clearStdin()
{
    while ((getchar()) != '\n');
}
```

```
  /*/////taille de la plateau*/
    scanf("%s",p);
    clearStdin();
    n=strtol(p,endptr,10);
    while((!isnumber(p))||(n==1))
    {
        printf("Taper un chiffre -> 1  \n");
        scanf("%s",p);
        clearStdin();
        n=strtol(p,endptr,10);
    }
    n=strtol(p,endptr,10);
    /*////plafont de la pile*/
    printf("\n*hauteur maximum de la pile = ");
    scanf("%s",chaut);
    clearStdin();
    while(!isnumber(chaut))
    {
        printf("Taper un chiffre\n");
        scanf("%s",chaut);
        while ((getchar()) != '\n');
    }
    haut=strtol(chaut,endptr,10);
```

4.2 Les variante séisme et l'affichage

J e pense que c'est le parti plus compliqué dans ce projet. À l'avance, j'ai met + autant que le curseur pour le choix, mais maintenance, on doit ajouter les pe-

tits étoiles * pour l'affichage du séisme. Si j'utilise le mécanisme comme l'avance c'est trop compliqué. Donc j'utilise le tableau plus grand pour l'affichage du séisme. On crée un tableau avec la taille comme ça.

$$la\ taille\ de\ plateau = 3n \times 3n \times haute$$

C 'est-à-dire on peut mettre les étoiles dans le plateau aussi et c'est facile de montrer les notations + et * et on peut seulement modifier le contenu dans le tableau et laisser les espaces qui sont les entourages du jeton être + ou *. C'est plus pratique, mais le reste de fonction veut manipuler le tableau avec la taille originale. Donc on va seulement élargir le plateau seulement dans cette fonction d'affichage.

```
/*@require: p et plateau pas NULL , haut > 0, n > 1
  @assigns: modifier 'p' et 'plateau' selon la resultat du seisme
  @ensure : retourne rien
 */
  void faire_seisme(char*** p,char*** plateau,int n,int haut)
{
    int x,y,z;
   /*///////clear////////*/
    for(x=0;x<3*n;x++)
        for(y=0;y<3*n;y++)
            for(z=0;z<haut;z++)
                p[x][y][z]='␣';
    /*////// copy ///////*/
    for(x=0;x<n;x++)
        for(y=0;y<n;y++)
            for(z=0;z<haut;z++)
                p[2*x+1][2*y+1][z]=plateau[x][y][z];
    /*////// seisme  /////*/
    int h;
    int down;
    int rest;
    for(x=0;x<n;x++)
        for(y=0;y<n;y++)
            {
                /*/////tomper////////*/
                h=1+geth(plateau,haut,x,y);
                down=seisme(n,h);
               /* printf("\n down:%d ",down); */ /*pour test*/
                if(down>0)
                {
                    rest=h-down;
                    for(rest=h-down;rest<haut;rest++)
                        p[2*x+1][2*y+1][rest]='.';
```

```
34                          /*//////mettre etoile//////*/
35                          p[2*x+1][2*y+0][0]='*';
36                          p[2*x+2][2*y+0][0]='*';
37                          p[2*x+2][2*y+1][0]='*';
38                          p[2*x+2][2*y+2][0]='*';
39                          p[2*x+1][2*y+2][0]='*';
40                          p[2*x+0][2*y+2][0]='*';
41                          p[2*x+0][2*y+1][0]='*';
42                          p[2*x+0][2*y+0][0]='*';
43                      }
44                  }
45      /*//////dupliquer a petit plateau encore////*/
46       for(x=0;x<n;x++)
47           for(y=0;y<n;y++)
48               for(z=0;z<haut;z++)
49                   plateau[x][y][z]=p[2*x+1][2*y+1][z];
50
51  }
52  /*@require: p et plateau pas NULL , haut > 0, n > 1
53    @assigns: modifier 'p' et 'plateau' selon la resultat du seisme
54    @ensure : l'affichage du seisme
55   */
56  void afficheChoisirSeisme(char*** p,char*** plateau,int n,int haut,int line_p,in
57  {
58     /*system("clear");*/
59     int x,y,z,i,toit;
60     char show;
61       /*///////clear le plateau ////////*/
62      for(x=0;x<3*n;x++)
63           for(y=0;y<3*n;y++)
64               if(p[x][y][1]=='+')
65                   p[x][y][1]=' ';
66      /*////// copy ////////*/
67      for(x=0;x<n;x++)
68           for(y=0;y<n;y++)
69               for(z=0;z<haut;z++)
70                   p[2*x+1][2*y+1][z]=plateau[x][y][z];
71     /*////mettre '+'/////*/
72     p[line_p][colonne_p+1][1]='+';
73     p[line_p][colonne_p-1][1]='+';
74     p[line_p+1][colonne_p][1]='+';
75     p[line_p-1][colonne_p][1]='+';
76     for(x=0;x<3*n;x++)
77     {
78        printf("\n");
79        for(y=0;y<3*n;y++)
```

```
                {

                    show=p[x][y][0];
                    if((p[x][y][0]=='*')&&(p[x][y][1]=='+'))
                        show='+';
                    if((p[x][y][0]!='*')&&(p[x][y][1]=='+'))
                        show='+';
                    if((p[x][y][0]=='.')||(p[x][y][0]=='X')||(p[x][y][0]=='O'))
                    {
                        show='.';
                        for(toit=0;toit<haut;toit++)
                            if(p[x][y][toit]!='.')
                                show=p[x][y][toit];
                    }


                    printf(" %c ",show);
                }

        }

        printf("\n");
        printf("\t\tle pile basse->haut : ");
            for(i=haut-1;i>-1;i--)
            {   printf("\n");
                printf("\t\t\t\t          %c ",p[line_p][colonne_p][i]);
            }

        printf("\n- - - - - - - - - - - - - - - - - - - - - - -\n\n\n");

}
```

4.3 Espace du mémoire

D ans ce projet, je choisis le tableau pour la structure de la donnée. Le taux d'utilisation n'est pas très haut, mais on peut manipuler le plateau et effectuer les actions plus rapides. Mais si la taille et la hauteur sont assez grandes la structure du tableau est tout à fait inapproprié.

5 Conclusion

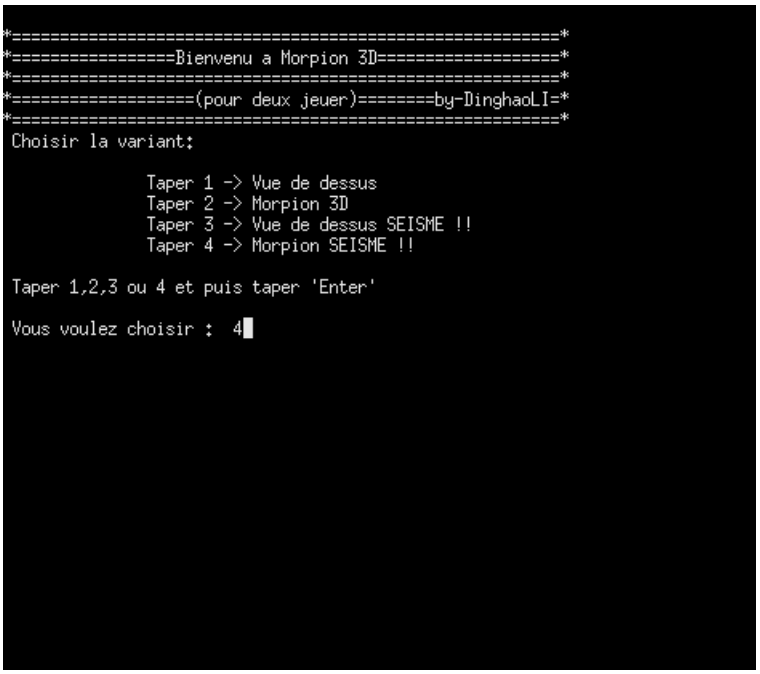
Pendant ce sujet, cette tâche m'a été d'un grand profit.

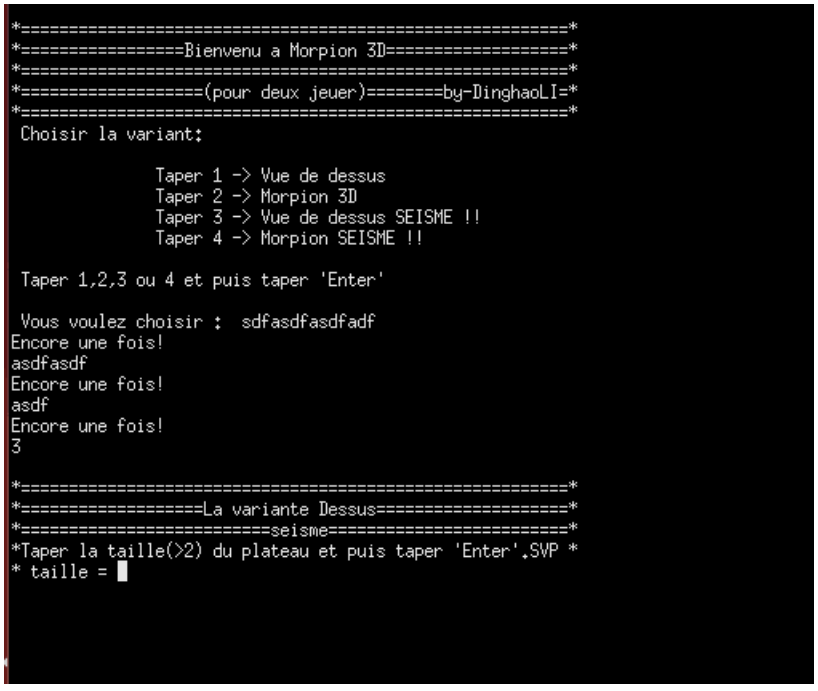
En fait, cet projet me donne une expérience très pratique et je peux obtenir l'habileté technique sur la programmation dans un projet concret. Je trouve que la modularisation et le commentaire sont très importants pour un projet

et tous les travailleurs. Il peut laisser les codes plus lisibles et plus faciles pour partager et communiquer avec les autres.

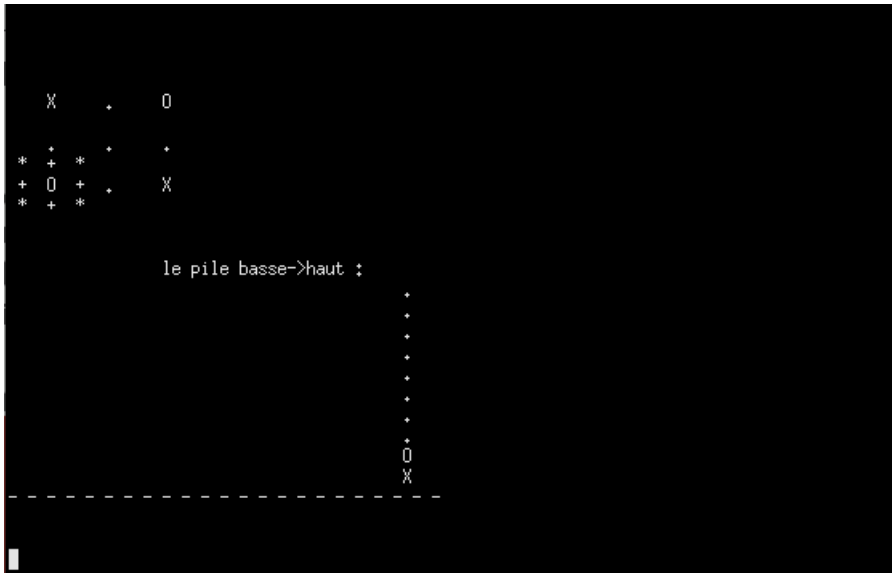
Et puis les algorithmes deviennent être en parfaite adéquation avec la structure de la donnée. Les deux sont très importants pour un projet aussi. En même temps durant on garantit les expériences de client, on doit traiter la relation entre la performance et la demande de client proprement.

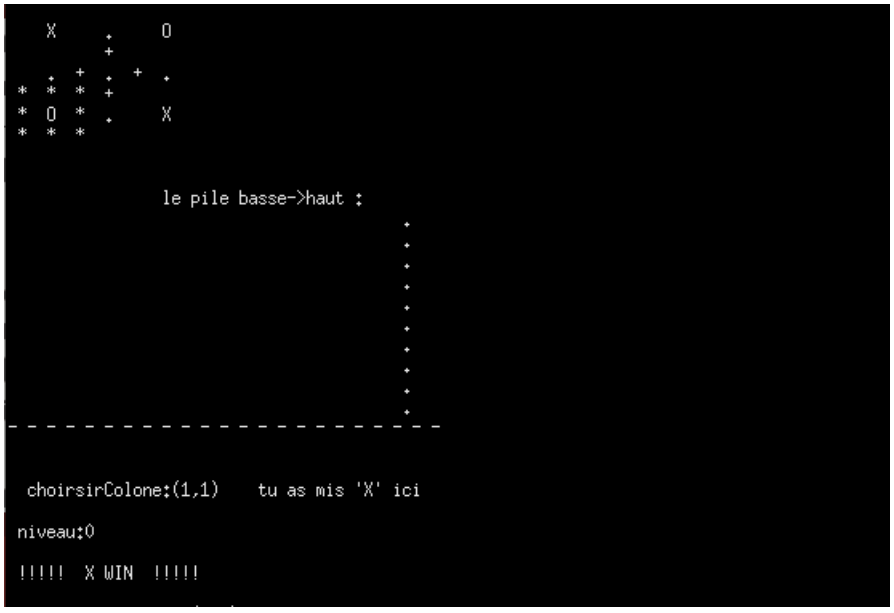
6 L'annexe


==========Bienvenu a Morpion 3D==========
==========(pour deux jeuer)========by-DinghaoLI=
Choisir la variant:
Taper 1 -> Vue de dessus
Taper 2 -> Morpion 3D
Taper 3 -> Vue de dessus SEISME !!
Taper 4 -> Morpion SEISME !!
Taper 1,2,3 ou 4 et puis taper 'Enter'
Vous voulez choisir : 4


==========Bienvenu a Morpion 3D==========
==========(pour deux jeuer)========by-DinghaoLI=
Choisir la variant:
Taper 1 -> Vue de dessus
Taper 2 -> Morpion 3D
Taper 3 -> Vue de dessus SEISME !!
Taper 4 -> Morpion SEISME !!
Taper 1,2,3 ou 4 et puis taper 'Enter'
Vous voulez choisir : sdfasdfasdfadf
Encore une fois!
asdfasdf
Encore une fois!
asdf
Encore une fois!
3
==========La variante Dessus==========
==========seisme==========
*Taper la taille(>2) du plateau et puis taper 'Enter'.SVP *
* taille =


le pile basse->haut :
choirsirColone:(0,0) tu as mis 'X' ici
line 1 colonne 1 diag 2 diag2 0
0 tour !!!
0 tour: p:ajouter r:retirer q:quitter : r


le pile basse->haut :


X O
X
O X
le pile basse->haut :
choirsirColone:(1,1) tu as mis 'X' ici
niveau:0
!!!!! X WIN !!!!!