Projet 5 : Combat naval

Henry Ledoux

7 Décembre 2001

Problème

Il est demandé de réaliser une implémentation en C++ d'un jeu de combat naval humain contre ordinateur. L'ordinateur choisis une grille parmis une panoplie prédéfinie, tandisque l'utilisateur doit entrer l'emplacement de ses 5 bateaux (respectivement de 5, 4, 3, 3 et 2 cases). L'ordinateur doit controler l'emplacement de chacun des bateaux de l'utilisateur avant de valider la grille.

A chaque tour de jeu, l'utilisateur entre les coordonées du point ou il veut tirer, l'ordinateur lui choisis une case au hasard jusqu'a ce qu'il touche un bateau, il cherche ensuite autour du point qu'il a touché, et enfin bombarde dans les deux directions.

La partie se termine lorsque tous les bateaux du joueur ou de l'ordinateur sont touchés.

Résolution

Pour y parvenir de façon simple, on crée principalement trois classes, une classe JoueurHumain, qui s'occupe de gérer le jeu de l'utilisateur et son interface; une classe JoueurVirtuel, qui elle s'occupe du jeu de l'ordinateur, elle contiendra le même type de fonctions, mais avec un contenu différent car les coups joués seront choisis par l'ordinateur. La troisieme classe est la classe jeu, qui appelle successivement ces fonctions et échange les "coups" entre l'utilisateur et l'ordinateur. Elle joue en quelque sorte le role d'arbitre.

Les classes Joueur contiennent comme données les deux plans de jeu : l'un contenant les bateaux du joueur et les coups destinés à ceux-ci, l'autre contenant uniquement les coups tirés contre l'adversaire, et un vecteur d'entiers de longueur 5, contenant le nombre de cases restantes pour chaque bateau, ceci afin de permettre de vérifier facilement si un bateau est coulé ou pas. Ce vecteur est aussi utilisé lors du placement des bateaux (dans le cas du joueur humain) car il est initialisé à la longueur de départ des bateaux et n'est pas encore modifié.

Ces classes contiennent diverses méthodes leur permettant de communiquer :

- tir(**int** &pos_x, **int** &pos_y)qui renvoie par référence le coup choisi par l'un ou l'autre camp;
- **int** recevoir_tir(**int** pos_x, **int** pos_y) qui prends en parametre les coordonées du coup tiré et reretourne un entier qui représente le résultat (0 touché, 1 coulé, 2 a l'eau);
- **void** donner_resultat(**int** result, **int** pos_x, **int** pos_y) sert à informer le tireur du résultat de son coup et enfin
- **bool** is_ended() qui renvoie *true* lorsque le joueur a perdu.

Toutes ces fonctions sont appelées a chaque tour par un élément de la classe Jeu selon l'exemple suivant :

```
int pos_x, pos_y, res;
User.tir(pos_x, pos_y);
res = Machine.recevoir_tir(pos_x, pos_y);
User.donner_resultat(res, pos_x, pos_y);
```

Vérifications

Afin de vérifier le bon fonctionnement du programme, il a falu reproduire certains cas limites, principalement pour ce qui est du controle de l'emplacement des bateaux de l'utilisateur, et de la stratégie de l'ordinateur.

Par exemple, pour le controle des emplacements, il a falu tester l'entrée de bateaux en diagonale, le long de chaque bord, de bateaux l'un contre l'autre. Pour la stratégie de l'ordinateur, les premiers tirs étant en partie aléatoires, il faut réessayer de facon a obtenir que le bateau soit touché sur un bout, ou encore que le bateau touché sois sur un bord du jeu.

Tous ces tests se sont avérés positifs.

Mode d'utilisation du programme

Lors de l'affichage des plans de jeu, la grille de gauche représente les bateaux de l'utilisateur avec les coups provenants de l'ordinateur, celle de droite représente les coups tirés par l'utilisateur.

En premier lieu, l'ordinateur demande d'entrer les extrémités de chacun des bateaux. Il faut entrer la lettre représentant la ligne suivie d'un numéro de 1 à 10 représentant la collonne. Si les positions entrées sont permises, l'ordinateur affichera les grilles avec le premier bateau placé, sinon il redemande les coordonées du bateau jusqu'à ce qu'elles soient correctes.

Une fois cette premiere étape terminée, il faut entrer de la meme maniere (lettre puis chiffre) les coordonées du point où tirer, l'ordinateur affiche ensuite le résultat du tir de l'utilisateur et de son tir a lui, modifie les grilles et affiche le résultat a l'écran.

A la fin de la partie, l'ordinateur affiche un message désignant le gagnant.

Le code

```
/* Projet 5: Combat Naval
 * Par: Henry Ledoux (Groupe 5)
 */
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
#include "grilles.h"
#include <dos.h> // Necessaire pour la fonction sleep()


// Converti le ècaractre éentr pour édsigner la ligne (de A a J) en entier (deà 0  9).
int char2int(char c)
{
  int res;
  if (c >= 'a' && c <= 'j')
     res = (int)(c - 'a');
  else if (c >= 'A' && c <= 'J')
     res = (int)(c - 'A');
  return res;
}

// Trie a et b, tel que a est plus petit que b.
void minitri(int &a, int &b)
{
   int Save;
   if (a > b)
   {
     Save = a;
     a = b;
     b = Save;
   }
}

/* La classe JoueurHumain propose diverses fonctions appelable par le corps de jeu. tir renvoie
   les écoordones éentre par l'utilisateur lorsqu'il édsire tirer contre l'ordinateur.
   recevoir_tir prends en parametre les écoordonnes du tir éexecut par l'ordinateur, et renvoie
   le érsultat: 0 étouch, 1 écoul, 2 dans l'eau.
   affiche() affiche les deux grilles de l'utilisateur a lé'cran. place_bateaux demande a
   l'utilisateur l'emplacement de chacun de ses bateaux. */

class JoueurHumain
{
  public:
    JoueurHumain();                                                    // 1
    void affiche();                                                    // 2
```

```
    void place_bateaux();                                                    // 3
    int recevoir_tir(int pos_x, int pos_y);                                  // 4
    void tir(int &pos_x, int &pos_y);                                        // 5
    void donner_resultat(int result, int pos_x, int pos_y);                  // 6
    bool is_ended();                                                         // 7

  private:
    void posebateau(int pos_a_x, int pos_a_y, int pos_b_x, int pos_b_y, int type);   // 8
    int check_boat(int pos_a_x, int pos_a_y, int pos_b_x, int pos_b_y, int type);    // 9
    plan_de_jeu gr_bateaux;
    plan_de_jeu gr_tirs;
    int reste_boat[5];
};

/* La classe JoueurVirtuel propose les memes fonctions, mais d'un contenu édiffrent,
   la fonction tir choisis automatiquement les écoordones. */
class JoueurVirtuel
{
   public:
      JoueurVirtuel();                                                       // 10
      void tir(int &pos_x, int &pos_y);                                      // 11
      int recevoir_tir(int pos_x, int pos_y);                                // 12
      void donner_resultat(int result, int pos_x, int pos_y);                // 13
      bool is_ended();                                                       // 14

   private:
      bool check_tir(int pos_x, int pos_y);                                  // 15
      plan_de_jeu gr_bateaux;
      plan_de_jeu gr_tirs;
      short int reste_boat[5];
      short int dir;
      short int etape;
      short int cur;
      short int tir_p_y;
      short int tir_p_x;
};

/* La classe jeu joue le role d'arbitre, elle s'occupe de gere les coups entre
   l'utilisateur et l'ordinateur */
class Jeu
{
   public:
      void tour();                                                           // 16
      void init();                                                           // 17
   private:
      void tir_user_mach();                                                  // 18
      void tir_mach_user();                                                  // 19
      JoueurHumain User;
      JoueurVirtuel Machine;
};

// 1 : éDfinitions des émthodes de la classe JoueurHumain, gere le jeu de l'utilisateur

JoueurHumain::JoueurHumain()
{
  for (int i = 0; i < 10; ++i)
    for (int j = 0; j < 10; ++j)
      gr_bateaux[i][j] = 0;

  for (int i = 0; i < 10; ++i)
    for (int j = 0; j < 10; ++j)
      gr_tirs[i][j] = 0;
```

```
  reste_boat[0] = 5;
  reste_boat[1] = 4;
  reste_boat[2] = 3;
  reste_boat[3] = 3;
  reste_boat[4] = 2;
}

// 2 : Fonction d'affichage des grilles utilisateur
void JoueurHumain::affiche()
{

    cout << char(0xDA) << char(0xC4);
        for (int i = 0; i < 10; ++i)
          cout << char(0xC2) << char(0xC4);
                cout << char(0xBF) << "    ";
      cout << char(0xDA) << char(0xC4);
        for (int i = 0; i < 10; ++i)
          cout << char(0xC2) << char(0xC4);
                cout << char(0xBF) << endl;

        cout << char(0xB3) << '#';
        for (int i = 0; i < 10; ++i)
                cout << char(0xB3) << (i+1)%10;
        cout << char(0xB3) << "    ";
      cout << char(0xB3) << '@';
        for (int i = 0; i < 10; ++i)
                cout << char(0xB3) << (i+1)%10;
        cout << char(0xB3);
    cout << endl;
        for (int i = 0; i < 10; ++i)
   {
      cout << char(0xC3) << char(0xC4);

        for (int j = 0; j < 10; ++j)
         cout << char(0xC5) << char(0xC4);
           cout << char(0xB4) << "    " << char(0xC3) << char(0xC4);

        for (int j = 0; j < 10; ++j)
         cout << char(0xC5) << char(0xC4);
           cout << char(0xB4) << endl << char(0xB3) << char(i+'A');

        for (int j = 0; j < 10; ++j)
      {
         cout << char(0xB3);
         if (gr_bateaux[i][j] == 0)
                cout << ' ';
         else if (gr_bateaux[i][j] == 6)
                cout << '~';
         else if (gr_bateaux[i][j] == 7)
                cout << '*';
         else
                cout << gr_bateaux[i][j];
      }

        cout << char(0xB3) << "    " << char(0xB3) << char(i+'A');

      for (int j = 0; j < 10; ++j)
      {
         cout << char(0xB3);
         if (gr_tirs[i][j] == 0)
                cout << ' ';
         else if (gr_tirs[i][j] == 1)
```

```
                cout << '*';
            else if (gr_tirs[i][j] == 2)
                cout << '~';
            else
                cout << gr_tirs[i][j];
        }

        cout << char(0xB3) << endl;
    }
    cout << char(0xC0) << char(0xC4);

    for (int j = 0; j < 10; ++j)
        cout << char(0xC1) << char(0xC4);
    cout << char(0xD9) << "    " << char(0xC0) << char(0xC4);
        for (int j = 0; j < 10; ++j)
        cout << char(0xC1) << char(0xC4);
        cout << char(0xD9) << endl;
}

// 3 : Demande a l'utilisateur l'emplacement de ses bateaux
void JoueurHumain::place_bateaux()
{
  for (int i = 1; i <= 5; ++i)
  {
    char c_temp;
    bool recommencer;
    int pos_a_x, pos_a_y, pos_b_x, pos_b_y;
    do
    {
      affiche();
      sleep(1);
      cout << "Entrez les deux extremites du ";
      switch (i)
      {
        case 1:
          cout << "porte-avions";
          break;
        case 2:
          cout << "cuirasse";
          break;
        case 3:
          cout << "croiseur";
          break;
        case 4:
          cout << "sous-marin";
          break;
        case 5:
          cout << "destroyer";
      };
                cout << " (longueur:" << reste_boat[i] << ")";
      cout << ". (Entrez la ligne sous forme de lettre, la colonne sous forme de chiffre)"
           << endl << "Premiere extremite: ";
      cin >> c_temp >> pos_a_y;
      pos_a_x = char2int(c_temp);
      pos_a_y--;
      cout << "Seconde extremite du bateau: ";
      cin >> c_temp >> pos_b_y;
      pos_b_x = char2int(c_temp);
      pos_b_y--;
      switch (check_boat(pos_a_x, pos_a_y, pos_b_x, pos_b_y, i))
      {
        case 0:
```

```
          cout << "ok" << endl;
          recommencer = false;
          break;
        case 1:
          cout << "Il faut entrer des coordonnees correctes" << endl;
          recommencer = true;
          break;
        case 2:
          cout << "Un bateau doit etre soit horizontal soit vertical!!" << endl;
          recommencer = true;
          break;
        case 3:
          cout << "Longueur du bateau incorrecte" << endl;
          recommencer = true;
          break;
        case 4:
          cout << "Position du bateau incorrecte" << endl;
          recommencer = true;
      };
    sleep(1);
    } while (recommencer); // L'ordinateur redemandera l'eplacement
                           // du bateau tant qu'il n'est pas correct.
    posebateau(pos_a_x, pos_a_y, pos_b_x, pos_b_y, i);
  }
}

// 4 : Recois les écoordones du tir de l'ordinateur, modifie la
//     grille utilisateur et renvoie le érsultat.
int JoueurHumain::recevoir_tir(int pos_x, int pos_y)
{
        int result;
   if (gr_bateaux[pos_x][pos_y] == 0)
   {
        result = 2;
                gr_bateaux[pos_x][pos_y] = 6;
        }
   else
   {
        reste_boat[gr_bateaux[pos_x][pos_y] - 1]--;
      if (reste_boat[gr_bateaux[pos_x][pos_y] - 1] == 0)
        result = 1;
      else
        result = 0;
      gr_bateaux[pos_x][pos_y] = 7;
   }
        return result;
}

// 5 : Demande a l'utilisateur les écoordones de l'endroit vers lequel il veut tirer
void JoueurHumain::tir(int &pos_x, int &pos_y)
{
        char temp;
        affiche();
   sleep(2);
   cout << "Entrez les coordonees de l'endroit ou vous voulez tirer" << endl;
   cin >> temp >> pos_y;
   pos_x = char2int(temp);
   pos_y--;
}
```

```
// 6 : Affiche le érsultat a lé'cran, modifie la grille de tir
//     et diminue le nombre de cases restante du bateau.
void JoueurHumain::donner_resultat(int result, int pos_x, int pos_y)
{
        if (result == 2)
   {
        cout << "Pas de chance! Dans l'eau.";
      gr_tirs[pos_x][pos_y] = 2;
   }
   else if (result == 0)
   {
        cout << "Bien! éTouch.";
                gr_tirs[pos_x][pos_y] = 1;
   }
   else
   {
        cout << "Youhoo! éCoul.";
      gr_tirs[pos_x][pos_y] = 1;
   }

}

// 7 : Cette fonction retourne vrai si le joueur a perdu.
bool JoueurHumain::is_ended()
{
        int tot = 0;
   for (int i = 0; i < 5; ++i)
        tot += reste_boat[i];
   return (tot == 0);
}

// 8 : Fonction éprive, éappele par place_bateaux(), place un bateau sur la grille utilisateur.
void JoueurHumain::posebateau(int pos_a_x, int pos_a_y, int pos_b_x, int pos_b_y, int type)
{
        minitri(pos_a_x, pos_b_x);
   minitri(pos_a_y, pos_b_y);
   for (int i = pos_a_x; i <= pos_b_x; ++i)
        for (int j = pos_a_y; j <= pos_b_y; ++j)
                gr_bateaux[i][j] = type;
}

// 9 : éVrifie que le placement du bateau soit éautoris avant de le placer.
int JoueurHumain::check_boat(int pos_a_x, int pos_a_y, int pos_b_x, int pos_b_y, int type)
{
  minitri(pos_a_x, pos_b_x);
  minitri(pos_a_y, pos_b_y);
  int erreur_nb = 0;
  if (pos_a_x > 9 || pos_a_x < 0 || pos_a_y > 9 || pos_a_y < 0 ||
                  pos_b_x > 9 || pos_b_x < 0 || pos_b_y > 9 || pos_b_y < 0)
    erreur_nb = 1;
  else if (pos_a_x != pos_b_x && pos_a_y != pos_b_y)
    erreur_nb = 2;
  else if (pos_a_x == pos_b_x && abs(pos_a_y - pos_b_y) != (reste_boat[type-1] - 1))
    erreur_nb = 3;
  else if (pos_a_y == pos_b_y && abs(pos_a_x - pos_b_x) != (reste_boat[type-1] - 1))
    erreur_nb = 3;
  else if ((pos_a_x == pos_b_x) && (pos_a_x == 0 || pos_b_x == 9))
    erreur_nb = 4;
  else if ((pos_a_y == pos_b_y) && (pos_a_y == 0 || pos_b_y == 9))
    erreur_nb = 4;
  else
  {
```

```
    bool bonne_pos = true;
    if (pos_a_x == 0)
        pos_a_x++;
    else if (pos_b_x == 9)
      pos_b_x--;
    if (pos_a_y == 0)
      pos_a_y++;
    else if (pos_b_y == 9)
      pos_b_y--;
    for (int i = pos_a_x - 1; i <= pos_b_x + 1; ++i)
        for (int j = pos_a_y - 1; j <= pos_b_y + 1; ++j)
        if ((i >= pos_a_x && i <= pos_b_x) || (j >= pos_a_x && i <= pos_b_x))
            bonne_pos = (bonne_pos && (gr_bateaux[i][j] == 0));
    if (!bonne_pos)
      erreur_nb = 4;
  }
  return erreur_nb;

}

// 10 : éDfinitions émthodes classe JoueurVirtuel

JoueurVirtuel::JoueurVirtuel()
{
        int choix_grille;
   srand(time(0));
   choix_grille = int(double(rand())/RAND_MAX*5);
   for (int i = 0; i < 10; ++i)
        for (int j = 0; j < 10; ++j)
      {
        gr_bateaux[i][j] = grilles[choix_grille][i][j];
                        gr_tirs[i][j] = 0;
      }
        reste_boat[0] = 5;
        reste_boat[1] = 4;
        reste_boat[2] = 3;
        reste_boat[3] = 3;
        reste_boat[4] = 2;
   etape = 0;
   dir = 0;
}

// 11 : Cette fonction retourne les écoordones choisies par l'ordinateur.
void JoueurVirtuel::tir(int &pos_x, int &pos_y)
{
   srand(time(0));
   if (etape == 0)
   {
        do
        {
                pos_x = int(double(rand())/RAND_MAX*10);
                        pos_y = int(double(rand())/RAND_MAX*10);
         tir_p_x = pos_x;
         tir_p_y = pos_y;
        } while (check_tir(pos_x, pos_y));
   }
   else if (etape == 1)
   {
                do
      {
        dir = (dir + 1) % 4;
        switch (dir)
```

```
        {

                case 0:
                pos_x = tir_p_x - 1;
             pos_y = tir_p_y;
             break;
          case 1:
                        pos_y = tir_p_y - 1;
             pos_x = tir_p_x;
                break;
          case 2:
                pos_x = tir_p_x + 1;
             pos_y = tir_p_y;
                        break;
                case 3:
                                pos_y = tir_p_y + 1;
                   pos_x = tir_p_x;
        };

     } while (check_tir(pos_x,pos_y));
   }
   else if (etape >= 2)
   {
        do
     {
                        if (etape == 2)
         {
                        cur++;
         }
        else
                        cur--;
                        if (dir == 0 || dir == 2)
                {
                        pos_x = tir_p_x + cur;
                                pos_y = tir_p_y;
                        }
                        else
                {
                pos_x = tir_p_x;
                pos_y = tir_p_y + cur;
        }
                        if (pos_x > 9 || pos_y > 9)
                etape = 3;
     } while (check_tir(pos_x,pos_y));
   }
}


// 12 : Recois le tir de l'utilisateur et renvoie le érsultat.
int JoueurVirtuel::recevoir_tir(int pos_x, int pos_y)
{
        int result;
   if (gr_bateaux[pos_x][pos_y] == 0)
   {
        result = 2;
        }
   else
   {
        reste_boat[gr_bateaux[pos_x][pos_y] - 1]--;
     if (reste_boat[gr_bateaux[pos_x][pos_y] - 1] == 0)
        result = 1;
     else
```

```
        result = 0;
    }
        return result;
}



// 13 : Modifie la grille de tir de l'ordinateur (afin de ne pas retirer au meme endroit).
void JoueurVirtuel::donner_resultat(int result, int pos_x, int pos_y)
{
    gr_tirs[pos_x][pos_y] = result + 1;
    if (result == 2)
        cout << "L'ordinateur a rate" << endl;
    else if (result == 0)
        cout << "L'ordinateur a touche" << endl;
    else
        cout << "L'ordinateur a coule" << endl;
    if (etape == 0 && result == 0)
    {
        etape = 1;
       dir = int(double(rand())/RAND_MAX*4);
    }
    else if (etape == 1 && result == 0)
    {
        cur = 0;
        etape = 2;
    }
    else if (etape == 2 && result == 3)
    {
        cur = 0;
        etape = 3;
    }
    else if (result == 1)
    {
        etape = 0;
    }
}

// 14 : Retourne vrai si l'ordinateur a perdu.
bool JoueurVirtuel::is_ended()
{
        int tot = 0;
    for (int i = 0; i < 5; ++i)
        tot += reste_boat[i];
    return (tot == 0);
}

// 15 : éVrifie que la valeur choisie par l'ordinateur est correcte
bool JoueurVirtuel::check_tir(int pos_x, int pos_y)
{
     return (gr_tirs[pos_x][pos_y] != 0 || pos_x < 0 || pos_x > 9 || pos_y < 0 || pos_y > 9);
}

// 16 : éDroulement du jeu
void Jeu::tour()
{
        while(!User.is_ended() && !Machine.is_ended())
    {
        tir_user_mach();
       if (!User.is_ended())
         tir_mach_user();
       sleep(2);
```

```
    }
    if (User.is_ended())
        cout << "Vous avez perdu!!" << endl;
    else
        cout << "Bravo! Vous avez gagne!!" << endl;
    sleep(2);
}

// 17 : Lance le jeu.
void Jeu::init()
{
        User.place_bateaux();
    tour();
}

// 18 : Tir de l'utilisateur contre la machine
void Jeu::tir_user_mach()
{
        int pos_x, pos_y, res;
    User.tir(pos_x, pos_y);
    res = Machine.recevoir_tir(pos_x, pos_y);
    User.donner_resultat(res, pos_x, pos_y);
}

// 19 : tir de la machine contre l'utilisateur
void Jeu::tir_mach_user()
{
        int pos_x, pos_y, res;
    Machine.tir(pos_x, pos_y);
    res = User.recevoir_tir(pos_x, pos_y);
    Machine.donner_resultat(res, pos_x, pos_y);
}

void main ()
{
        Jeu the_game;
    the_game.init();
    sleep(2);
}
```