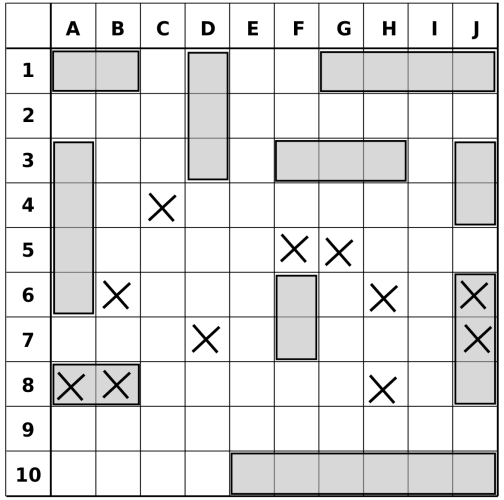


Mini Projet Bataille Navale

Bastien Lavaux - Lucas Dargere - Nirmine Kortam - Fatoumata Seye

Professeurs: Myriam Servières - Olivier Roux

Plan

- Présentation du mini-projet
- Conception
- Réalisation
- Déroulement de la partie

Présentation du projet

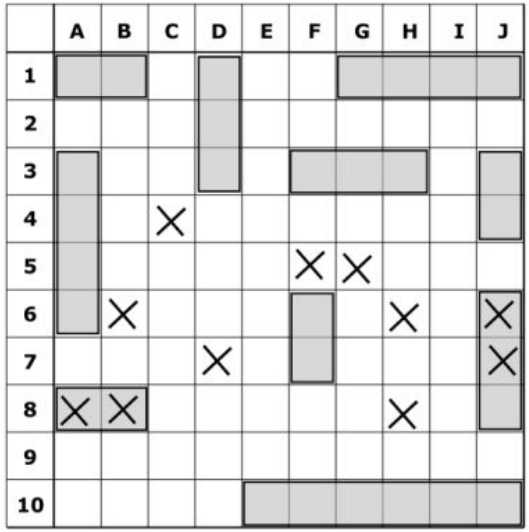


FIGURE 1.1 – Grille et bateaux

Chaque joueur dispose de deux grilles de 10x10 cases.

- Une pour la position de ses propres bateaux (1 porte-avion de 5 cases, 1 croiseur de 4 cases, 2 contre-torpilleurs de 3 cases chacun et 1 torpilleur de 2 cases)
- l'autre pour suivre les tirs effectués sur la flotte ennemie

-

Conception

Création de deux classes:

Classe **plateau**:

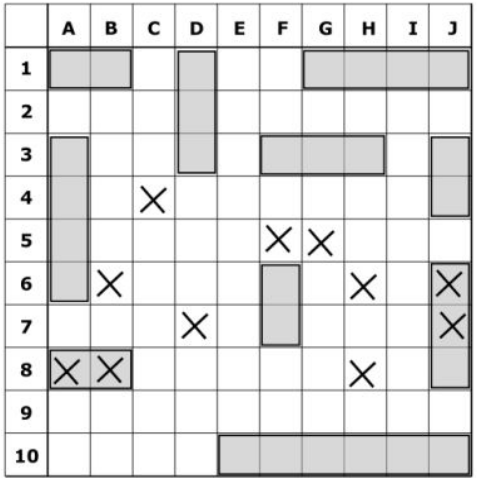


FIGURE 1.1 – Grille et bateaux

Diagramme de Classe **Plateau**:

plateau

- tableau: int[1][100]
- la_tire: int[1][100]
- flottevie: int
- porte_avion: bateau*
- croiseur: bateau*
- contre-torpilleur1: bateau*
- contre-torpilleur2: bateau*
- torpilleur: bateau*
- flotte: vector<bateau*>

+ initialisation() : void
+ afficherPlateau(): void
+ placer(int x, int y) : void
+ verifNotation(action: string): string
+ transfoChar(notation:string&): int
+ retourCoor(string message): int
+ tirer(int X, int Y): int
+ placementVerHaut (typebateau: Bateau*, coordbateau: int) : void
+ placementVerBas (typebateau: Bateau*, coordbateau: int) : void
+ placementHorzDroite(typebateau: Bateau*, coordbateau:int) : void
+ placementhorzGauche (typebateau: Bateau*, coordbateau:int) : void
+ verifcase(coordonées: int , etat: int) : bool
+ verifVerHaut(typebateau: Bateau*, coordbateau: int): bool
+ choix(typebateau :Bateau* , coordbateau: int): void
+ verifVerBas(typebateau :Bateau* , coordbateau: int): bool
+ verifVerHorzDroite(typebateau :Bateau* , coordbateau: int): bool
+ verifVerHorsGauche(typebateau: Bateau* , coordbateau: int): bool
+ bateautouche (position: int) bool
+ modifCase(position:int, etat:int):void
+ flottevivante(): bool
+ modifFlotteVie(): void
+ PlacementCaseIA (coordbateau : int, j : int , flottetour : int): void
+ placementIA(i : int) : void
+ tireIA(*def: Plateau) : void

Conception

Création de deux classes:

Classe **bateau**:

Diagramme de Classe **bateau**:

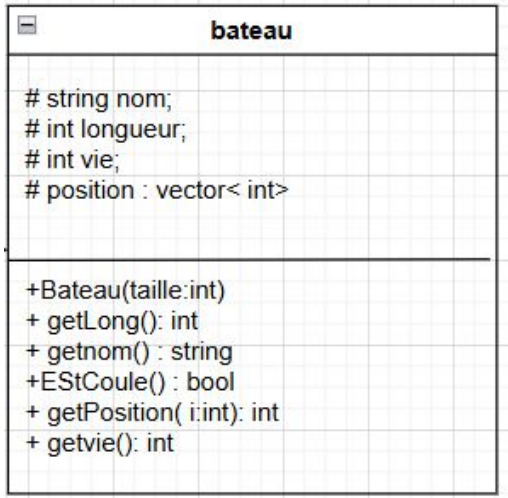


Diagramme de classe

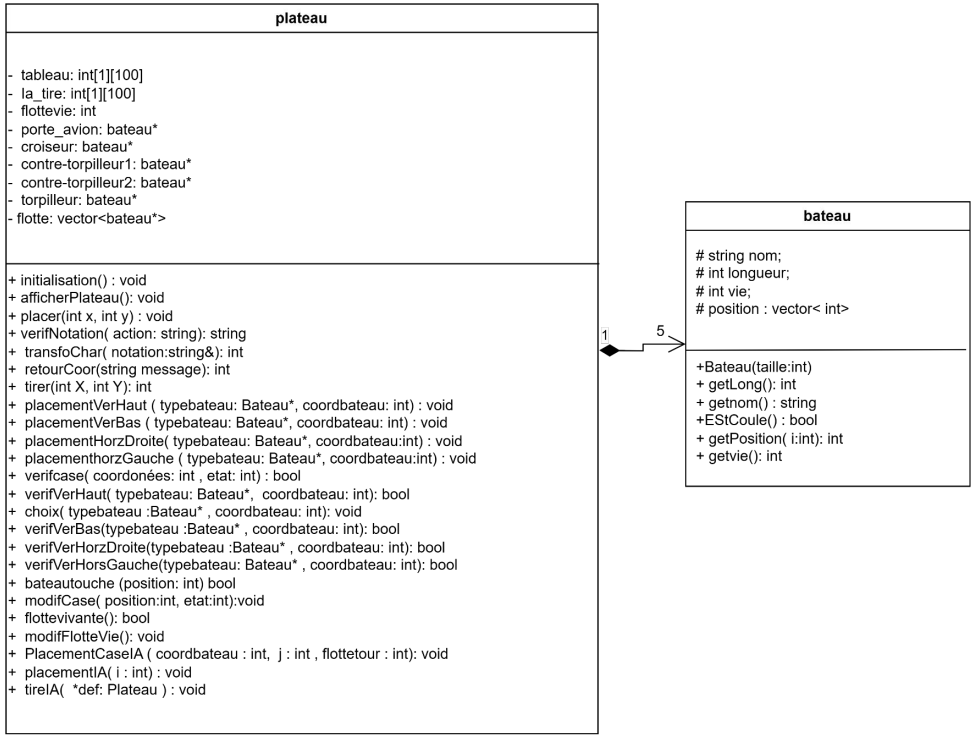
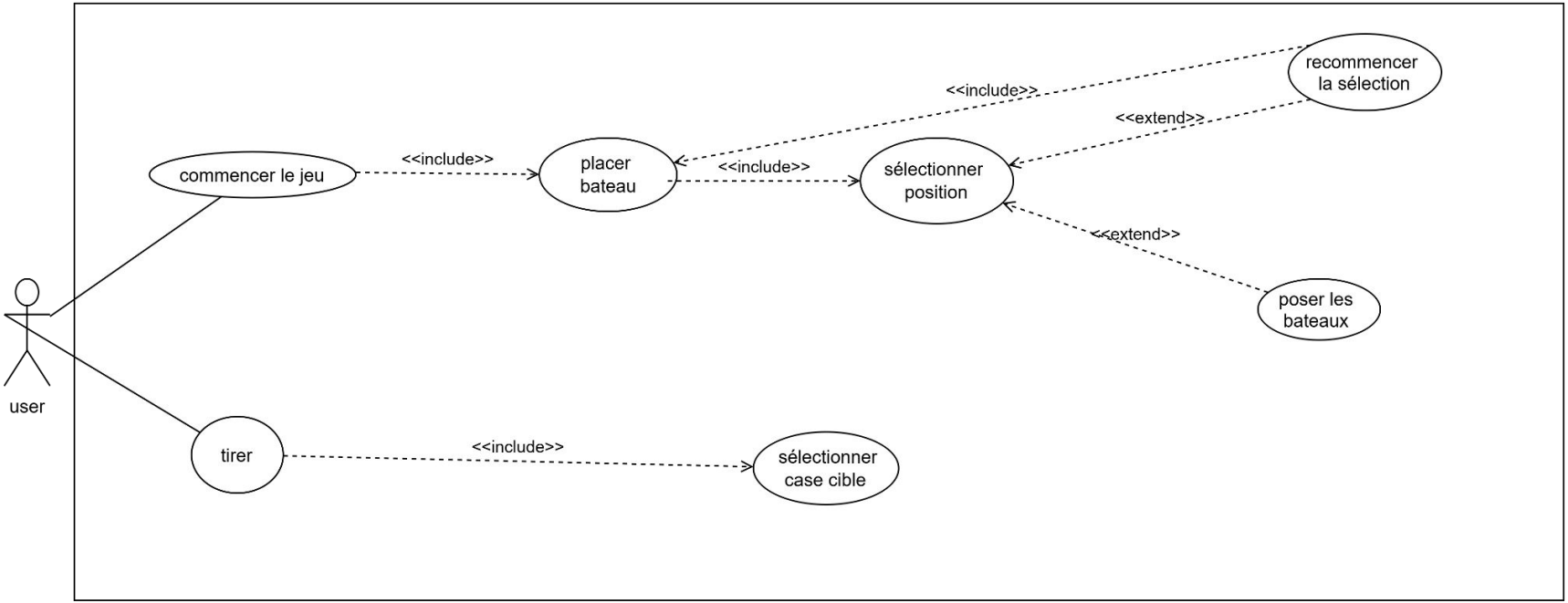


Diagramme d’utilisation:



Conception

Evolution des classes: La classe Case

```
class Case{

private:
  char x;
  int y;
  int status;

public:
//constructeur
  Case();
  Case(char x, int y, int status);

//getter
  char getx() const;
  int gety() const;
  int getstatus();

//setter
  void setx(char x);
  void sety(int y);
  void setstatus(int status);
};
```

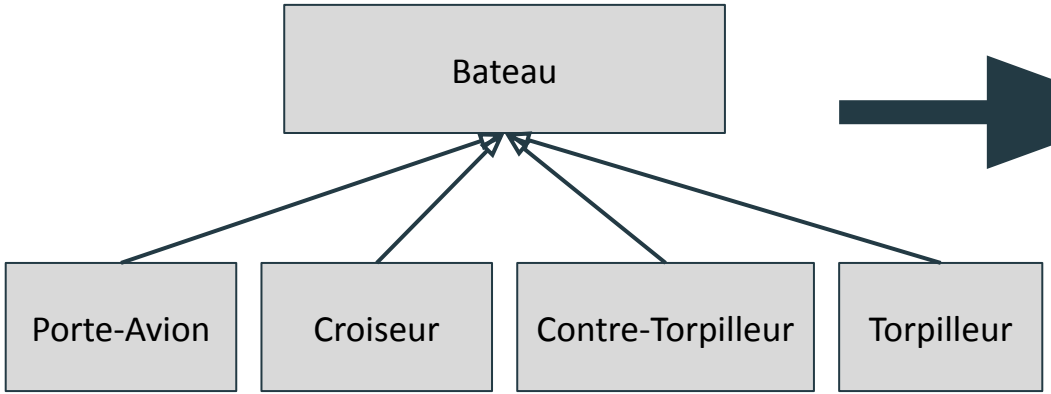
```
#define EAU 0
#define BATEAU 1
#define FAIL 2
#define TOUCHER_ATT 3
#define INTERDIT 4
#define TOUCHER_DEF 5
```

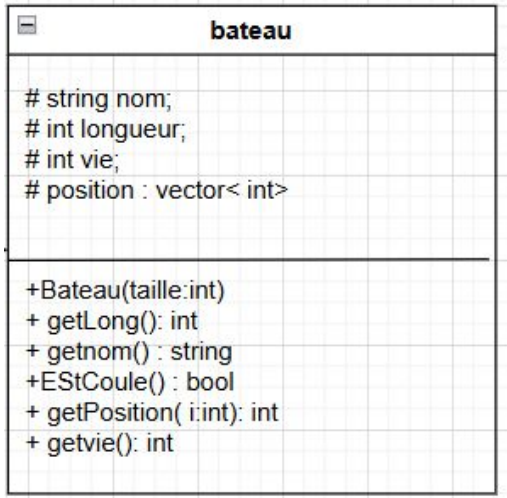
```
if (tableau[0][i] == INTERDIT){
  tableau[0][i] = EAU;
}
```

```
enum Etat {
    EAU,
    BATEAU,
    FAIL,
    TOUCHER_ATT,
    TOUCHER_DEF,
    INTERDIT
};
```

Conception

Evolution des classes: La classe Bateau





Conception

Codes intéressants: Le transcodage

```
int Plateau::transfoChar(string &notation) {
  int x = toupper(notation[0]);
  if (x == 'A') {
    x = 0;
  } else if (x == 'B') {
    x = 1;
  } else if (x == 'C') {
    x = 2;
  } else if (x == 'D') {
    x = 3;
  } else if (x == 'E') {
    x = 4;
  } else if (x == 'F') {
    x = 5;
  } else if (x == 'G') {
    x = 6;
  } else if (x == 'H') {
    x = 7;
  } else if (x == 'I') {
    x = 8;
  } else if (x == 'J') {
    x = 9;
  }
  x = x + 10 * (stoi(notation.substr(1))) - 10;
  return x;
}
```

A	B	C	D	E	F	G
1	2	3	4	5	6	7
H	I	J	K	L	M	N
8	9	10	11	12	13	14
O	P	Q	R	S	T	U
15	16	17	18	19	20	21
V	W	X	Y	Z		
22	23	24	25	26		

Conception

Codes intéressants: Le stockage des bateaux

```
  Bateau *porte_avion;
avion*/
  Bateau *croiseur;
croisseur*/
  Bateau *contre_torpilleur1;
torpilleur 1*/
  Bateau *contre_torpilleur2;
torpilleur 2*/
  Bateau *torpilleur;
torpilleur  */
  vector<Bateau *> flotte =
      {}; /** vecteur flotte
```

```
bool Plateau::bateautouche(int position) {

  for (int i = 0; i < porte_avion->getLong(); i++) {

    if (porte_avion->getPosition(i) == position) {
      if (0 == porte_avion->getvie() - 1) {
        return true;
      }
      porte_avion->diminuerVie();
    }
  }
  for (int i = 0; i < croiseur->getLong(); i++) {
    if (croiseur->getPosition(i) == position) {
      if (0 == croiseur->getvie() - 1) {
        return true;
      }
      croiseur->diminuerVie();
    }
  }
```

```
while (i < flotte.size()) {

  int coordbateau = dis2(gen);

  while (verifcase(coordbateau, 1) ||
erifcase(coordbateau, 4)) {

    coordbateau = dis2(gen);

    if (verifHorzDroite(flotte[i], coordbateau) == false
```

Conception

Codes intéressants: Méthode de vérification de cases libres

```
bool Plateau::verifHorzDroite(Bateau *typebateau, int coordbateau) {
  for (int y = 1; y <= typebateau->getLong() - 1; ++y) {
    if ((coordbateau / 10) != (coordbateau + y) / 10) {
      return false;
    } else if (tableau[0][coordbateau + y] == 1) {
      return false;

    } else if (tableau[0][coordbateau + y] == 4) {
      return false;
    }
  }
  return true;
}
```

```
#define EAU 0
#define BATEAU 1
#define FAIL 2
#define TOUCHER_ATT 3
#define INTERDIT 4
#define TOUCHER_DEF 5
```

Conception

Codes intéressants: Méthode choix - Interface joueur

```
void Plateau::choix(Bateau *typebateau, int coordbateau) {
  string verifchoix = "";
  string input = "";
  if (verifVerHaut(typebateau, coordbateau) == false &&
      verifVerBas(typebateau, coordbateau) == false &&
      verifHorzDroite(typebateau, coordbateau) == false &&
      verifHorzGauche(typebateau, coordbateau) == false) {
    cout << " vous ne pouvez pas positionner le bateau ici" << endl;
    placement(typebateau);
  } else {
    cout << "choissez l'orientation de votre bateau parmis les différents "
            "choix \n";
    if (verifVerHaut(typebateau, coordbateau)) {
      cout << "1: vers le haut \n";
      verifchoix = "1";
    };
    if (verifVerBas(typebateau, coordbateau)) {
      cout << "2: vers le bas \n";
      verifchoix = verifchoix + "2";
    };
    if (verifHorzDroite(typebateau, coordbateau)) {
      cout << "3: vers droite \n";
      verifchoix = verifchoix + "3";
    };
```

```
    verifchoix = verifchoix + "3";
  };
  if (verifHorzGauche(typebateau, coordbateau)) {
    cout << "4: vers gauche \n";
    verifchoix = verifchoix + "4";
  };
  string input;
  cin >> input;
```

Conception

Codes intéressants: Méthode choix - Interface joueur machine (suite)

```
string input;
cin >> input;
if (input.length() == 1) {
  for (int i = 0; i < verifchoix.length(); ++i) {
    if (input[0] == verifchoix[i]) {
      cout << "vous avez choisi l'orientation " << input << endl;

      if (input[0] == '1') {
        placementVerHaut(typebateau, coordbateau);
      } else if (input[0] == '2') {
        placementVerBas(typebateau, coordbateau);
      } else if (input[0] == '3') {
        placementHorzDroite(typebateau, coordbateau);

      } else if (input[0] == '4') {
        placementHorzGauche(typebateau, coordbateau);
      }
      return;
```

```
        return;
      }
    }
    cout << "vous n'avez pas renseigné un chiffre valable" << endl;
    choix(typebateau, coordbateau);
  } else {
    cout << "vous n'avez pas renseigné un chiffre valable" << endl;
    choix(typebateau, coordbateau);
  }
 }
}
```

Conception

Codes intéressants: tir de l'IA

```
bool Plateau::tireIA(Plateau *def) {
  std::random_device rd;
  std::mt19937 gen(rd());
  bool etat = false;

  vector<int> caselibre;
  int tour = 0;

  for (int i = 0; i < 101; ++i) {
    if (0 == Ia_tire[0][i]) {
      caselibre.push_back(i);
      tour++;
    }
  }
  if (toucher == 1) {
    int i = tracking(def);
    if (i == 0) {
    }
    if (i == 1)
      return true;
    if (i == 2) {
      return false;
    }
  }
  uniform_int_distribution<> dis1(0, tour);
  int coordonne = dis1(gen);

  for (int i = 0; i < caselibre.size(); ++i) {
    if (coordonne == caselibre[i]) {

      if (def->verifcase(coordonne, 1)) {

        etat = def->bateautouche(coordonne);
        def->modifCase(coordonne, 5);
        modifCase(coordonne, 3);
        Ia_tire[0][coordonne] = 1;
        if (etat) {
          cout << "Touché Coulé !" << endl;
          def->modifFlotteVie();
        } else {
          cout << "Touché !" << endl;
        }
        if (def->flottevivante() == false) {
          return false;
        }
      } else {
        modifCase(coordonne, 2);
        cout << "Manquer" << endl;
      }
    }
  }
  return true;
}
```

Conception

Codes intéressants: tracking de l'IA

```
698 int Plateau::tracking(Plateau *def) {
699   random_device rd;
700   mt19937 gen(rd());
701   vector<int> valid = {};
702   if (derniertire / 10 == (derniertire + 1) / 10 || derniertire + 1 >
    100) {
703     valid.push_back(derniertire + 1);
704     cout << valid[0] << endl;
705   }
706   if (derniertire / 10 == (derniertire - 1) / 10 || derniertire - 1 >
    0) {
707     valid.push_back(derniertire - 1);
708     cout << valid[1] << endl;
709   }
710   if (derniertire - 10 > 0) {
711     valid.push_back(derniertire - 10);
712     cout << valid[2] << endl;
713   }
714   if (derniertire + 10 < 100) {
715     valid.push_back(derniertire + 10);
716     cout << valid[3] << endl;
717   }
718   if (valid.size() == 0) {
719     return 0;
720   } else {
721     while (true) {
```

```
uniform_int_distribution<> dis1(0, valid.size() - 1);
cout << valid.size() << endl;
int choixc = dis1(gen);
cout << choixc << endl;
this_thread::sleep_for(chrono::milliseconds(100));
if (def->verifcase(valid[choixc], 1)) {
  bool etat = def->bateautouche(choixc);
  def->modifCase(valid[choixc], 5);
  modifCase(valid[choixc], 3);
  Ia_tire[0][valid[choixc]] = 1;
  if (etat) {
    cout << "Touché Coulé ! \n" << endl;
    def->modifFlotteVie();
    toucher = 0;
    derniertire = 0;
    cout << "traking désactivé \n" << endl;
  } else {
    cout << "Touché ! \n" << endl;
    toucher = 1;
    derniertire = valid[choixc];
    cout << "traking activé \n " << endl;
  }
```

Conception

Codes intéressants: tracking de l'IA

```
        if (def->flottevivante() == false) {
          return 2;
        } else {
          return 1;
        }
        break;
      } else if (verifcase(valid[choixc], 0)) {
        modifCase(valid[choixc], 2);
        cout << "Manquer \n" << endl;
        toucher = 0;
        cout << "traking désactivé \n" << endl;
        return 1;
      }
    }
  }
}
```

Pistes d’améliorations

- Harmonisations des programmations (fonctions Tirer() dans le main.cpp a mettre dans le plateau(), utilisations du vecteurs “flotte” au lieu de l’appel des bateau directement)
- Réécriture de certains morceaux du codes (les rounds inutiles, utilisations des #define)
- Faciliter les modifications avec des attributs au lieu de valeur inscrites dans le codes (LMAX au lieu de 100, …)
- Amélioration du tracking (tracking plus complet permettant de continuer à viser autour d’une case toucher si on rate le tir d’après)

Déroulement de la partie :

Merci de votre écoute