Compte Rendu TP3 CPOO

Maxime Havez, Gareth Thiveux

INSA de Rennes
4INFO, groupe 2.2

8 octobre 2010

Listing 1 – Fichier en-tête de la classe Joueur

```
1  // joueur.h
2
3
4
5  #ifndef JOUEUR_H
6  #define JOUEUR_H
7
8  #include <iostream>
9  using namespace std;
10
11 class Partie;
12 class Case;
13
14 class Joueur {
15 protected :
16   Partie * maPartie;
17 public:
18     Joueur(Partie *);
19     virtual void jouer();
20   virtual void effectuerCoupSurCase(Case * c)=0 ;
21   //classe virtual pure ! -> on ne peut pas l'instancier
22   virtual Case * choisirUneCase()=0;
23     void afficher();
24 };
25
26 #endif
```

Listing 2 – Classe Joueur

```
1  // joueur.cpp
```

```
2
3 #include "joueur.h"
4
5
6
7 Joueur::Joueur(Partie * p) {
8   maPartie = p;
9 }
10
11
12
13 void Joueur::jouer() {
14   cout << "Joueur::jouer(): debut:" << endl;
15   Case * c = choisirUneCase();
16   effectuerCoupSurCase(c);
17   cout << "Joueur::jouer(): fin." << endl;
18 }
19
20
21 void Joueur::afficher() {
22
23   cout << " ma Partie = ";
24   if (maPartie) cout << maPartie << endl;
25   else cout << " null..." << endl;
26 }
```

Listing 3 – Fichier en-tête de la classeAnge

```
1 // ange.h
2
3 #ifndef ANGE_H
4 #define ANGE_H
5
6 #include "joueur.h"
7
8 class Partie;
9
10 class Ange : public Joueur {
11
12 protected:
13     int puissance;
14   Case * maCase;
15 public:
16   Ange(Partie *, int);
17   void setCase(Case *c){maCase=c;}
18     bool jeSuisBloque();
19     bool jeSuisLibre();
20     void afficher();
21   virtual void Ange::effectuerCoupSurCase(Case * c);
22
```

```
23   bool caseInaccessible(Case * c);// renvoie vrai si l'ange ne peut atteindre la
       case
24 };
25
26 #endif
```

Listing 4 – Classe Ange

```
1  // ange.cpp
2
3  #include "ange.h"
4  #include "partie.h"
5
6
7
8  Ange::Ange(Partie * pa, int pui) : Joueur(pa) {
9    puissance = pui;
10 }
11
12 bool Ange::jeSuisBloque() {
13   bool uneCaseLibre = false;
14   int t = maPartie->monDamier->taille;
15   int i, j;
16   for (i=0; i<t; i++) {
17     for (j=0; j<t; j++) {
18       Case * c = maPartie->monDamier->mesCases[i][j];
19       if ( maCase->distance(c) <= puissance )
20   if (!(c->estBouchee()) && (!c->estAnge())) uneCaseLibre = true;
21     }
22   }
23   return !uneCaseLibre;
24 }
25
26 bool Ange::jeSuisLibre() {
27   if ((maCase->getX()==0) || (maCase->getY()==0)) return true;
28   int t = maPartie->monDamier->taille;
29   if ((maCase->getX()==t-1) || (maCase->getY()==t-1)) return true;
30   return false;
31 }
32
33 void Ange::afficher() {
34   Joueur::afficher();
35   cout << " puissance = " << puissance << endl;
36   cout << " ma Case = ";
37   if (maCase) cout << maCase << endl;
38   else cout << " null..." << endl;
39 }
40
41 void Ange::effectuerCoupSurCase(Case * c) {
42   maCase->setAnge(NULL);
```

```
43   maCase = maPartie->monDamier->mesCases[c->getX()][c->getY()];
44   maPartie->monDamier->mesCases[c->getX()][c->getY()]->setAnge(this);
45 }
46
47 bool Ange::caseInaccessible(Case * c) {
48   bool ok=false;
49   if (c->distance(maCase) > puissance){
50     cout << "Erreur: case inaccessible." << endl;
51     ok= true;
52   }
53   return ok;
54 }
```

Listing 5 – Fichier en-tête de la classe Diable

```
1 // diable.h
2
3 #ifndef DIABLE_H
4 #define DIABLE_H
5
6 #include "joueur.h"
7
8 class Partie;
9
10
11 class Diable : public Joueur {
12 public:
13   Diable(Partie *);
14   void afficher();
15   virtual void Diable::effectuerCoupSurCase(Case * c);
16 };
17
18 #endif
```

Listing 6 – Classe Diable

```
1 // diable.cpp
2
3 #include "diable.h"
4 #include "partie.h"
5
6 Diable::Diable(Partie * p) : Joueur(p) {
7 }
8
9
10 void Diable::afficher() {
11   Joueur::afficher();
12 }
13
14
```

```
15 void Diable::effectuerCoupSurCase(Case * c) {
16   maPartie->monDamier->mesCases[c->getX()][c->getY()]->setBouchee(true);
17 }
```

Listing 7 – Fichier en-tête de la classe AngeHumain

```
1  // angeHumain.h
2
3  #ifndef ANGE_HUMAIN_H
4  #define ANGE_HUMAIN_H
5
6
7  #include "ange.h"
8
9
10 class Partie;
11
12 class AngeHumain : public Ange {
13 public:
14   AngeHumain(Partie *, int);
15   void afficherPrompt();
16   /*void jouer();*/
17   Case * choisirUneCase();
18   /*void effectuerCoupSurCase(Case *);*/
19   bool verifier(int x,int y);
20  };
21
22 #endif
```

Listing 8 – Classe AngeHumain

```
1  // angeHumain.cpp
2
3
4
5  #include "angeHumain.h"
6
7  #include "partie.h"
8
9
10
11
12
13 AngeHumain::AngeHumain(Partie * pa, int pui) :
14
15 Ange(pa, pui) {
16
17 }
18
19
```

```
20
21 void AngeHumain::afficherPrompt() {
22
23   cout << "Ange > ";
24
25 }
26
27
28
29
30
31 /*void AngeHumain::jouer(){
32   cout << "Joueur::jouer(): debut:" << endl;
33   Case * c = choisirUneCase();
34   effectuerCoupSurCase(c);
35   cout << "Joueur::jouer(): fin." << endl;
36 }*/
37
38
39
40
41
42
43
44 Case * AngeHumain::choisirUneCase() {
45
46   int x, y;
47
48   bool ok;
49
50   Case * c;
51
52   // lutilisateur choisit un coup
53
54   do {
55
56     // lutilisateur tape un coup
57
58     cout << "Ange > x ? ";
59
60     cin >> x;
61
62     cout << "Ange > y ? ";
63
64     cin >> y;
65
66     // le programme verifie le coup
67
68     if(ok=verifier(x,y)){
```

```
69
70        c = maPartie->monDamier->mesCases[x-1][y-1];
71
72        ok=!(Ange::caseInaccessible(c));
73
74      }
75
76    } while (!ok);
77
78    return c;
79
80
81
82 }
83
84
85
86 /*void AngeHumain::effectuerCoupSurCase(Case * c) {
87    maCase->setAnge(NULL);
88    maCase = maPartie->monDamier->mesCases[c->getX()][c->getY()];
89    maPartie->monDamier->mesCases[c->getX()][c->getY()]->setAnge(this);
90 }*/
91
92
93 bool AngeHumain::verifier(int x,int y){
94
95    bool ok=true;
96
97    Case * c;
98
99    if ((x>0) && (y>0) &&
100
101     (x<=maPartie->monDamier->taille)&&
102
103     (y<=maPartie->monDamier->taille)) {
104
105       ok = true;
106
107       c = maPartie->monDamier->mesCases[x-1][y-1];
108
109       if (c->estBouchee()) {
110
111         cout << "Erreur: case bouchee." << endl; ok = false;
112
113       }
114
115       if (c->estAnge()) {
116
117         cout << "Erreur: case occupee par l'ange." << endl; ok = false;
```

```
118
119       }
120
121     }
122
123     else ok = false;
124
125   return ok;
126
127 }
```

Listing 9 – Fichier en-tête de la classe AngeAleatoire

```
1  // angeAleatoire.h
2
3  #ifndef ANGE_ALEATOIRE_H
4  #define ANGE_ALEATOIRE_H
5
6  #include "ange.h"
7  #include "alea.h"
8
9  class Partie;
10
11 class AngeAleatoire : public Ange {
12 public:
13   AngeAleatoire(Partie *, int);
14   /*void jouer();*/
15   Case * choisirUneCase();
16   /*void effectuerCoupSurCase(Case *);*/
17 };
18
19 #endif
```

Listing 10 – Classe AngeAleatoire

```
1  // angeAleatoire.cpp
2
3  #include "angeAleatoire.h"
4  #include "partie.h"
5
6
7
8  AngeAleatoire::AngeAleatoire(Partie * pa, int pui) :
9  Ange(pa, pui)
10 {
11 }
12
13 /*void AngeAleatoire::jouer(){
14   cout << "Joueur::jouer(): debut:" << endl;
15   Case * c = choisirUneCase();
```

```
16   effectuerCoupSurCase(c);
17   cout << "Joueur::jouer(): fin." << endl;
18 }*/
19
20 Case * AngeAleatoire::choisirUneCase() {
21   int x = 0;
22
23   int y = 0;
24
25   int t = maPartie->monDamier->taille;
26
27   int i, j, n, r;
28
29   Case * c;
30
31   // le programme choisit un coup
32
33   n = 0; // on compte le nombre de coups possibles.
34
35   for (i=0; i<t; i++) {
36
37     for (j=0; j<t; j++) {
38
39       c = maPartie->monDamier->mesCases[i][j];
40
41       if ((c->distance(maCase)<=puissance) && !(c->estBouchee()) &&
           (!c->estAnge()))
42
43         n++;
44
45     }
46
47   }
48
49   r = Alea::engendrer(n);
50
51   n = 0; // on selectionne un coup aleatoire dans les coups possibles.
52
53   for (i=0; i<t; i++) {
54
55     for (j=0; j<t; j++) {
56
57       c = maPartie->monDamier->mesCases[i][j];
58
59       if (!(Ange::caseInaccessible(c)) && !(c->estBouchee()) && (!c->estAnge()))
60
61         if (++n == r) {
62
63           x = c->getX() + 1;
```

```
64
65            y = c->getY() + 1;
66
67          }
68
69        }
70
71    }
72    return c;
73 }
74
75 /*void AngeAleatoire::effectuerCoupSurCase(Case * c) {
76    maCase->setAnge(NULL);
77    maCase = maPartie->monDamier->mesCases[c->getX()][c->getY()];
78    maPartie->monDamier->mesCases[c->getX()][c->getY()]->setAnge(this);
79 }*/
```

Listing 11 – Fichier en-tête de la classe DiableHumain

```
1  // diableHumain.h
2
3  #ifndef DIABLE_HUMAIN_H
4  #define DIABLE_HUMAIN_H
5
6  #include "diable.h"
7
8  class Partie;
9
10
11 class DiableHumain : public Diable{
12 public:
13    DiableHumain(Partie *);
14    void afficherPrompt();
15    /*void jouer();*/
16    Case * choisirUneCase();
17    /*void effectuerCoupSurCase(Case *);*/
18    bool verifier(int x,int y);
19 };
20
21 #endif
```

Listing 12 – Classe DiableHumain

```
1  // diableHumain.cpp
2
3
4
5  #include "diableHumain.h"
6
7  #include "partie.h"
```

```
8
9
10
11
12
13
14
15 DiableHumain::DiableHumain(Partie * p) :
16
17 Diable(p)
18
19 {}
20
21
22
23 void DiableHumain::afficherPrompt() {
24
25   cout << "Diable > ";
26
27 }
28
29
30
31 /*void DiableHumain::jouer(){
32   cout << "Joueur::jouer(): debut:" << endl;
33   Case * c = choisirUneCase();
34   effectuerCoupSurCase(c);
35   cout << "Joueur::jouer(): fin." << endl;
36 }*/
37
38
39
40 Case * DiableHumain::choisirUneCase() {
41
42   int x, y;
43
44   bool ok;
45
46   Case * c;
47
48   // lutilisateur choisit un coup
49
50   do {
51
52   // lutilisateur tape un coup
53
54     cout << "Diable > x ? ";
55
56     cin >> x;
```

```
57
58      cout << "Diable > y ? ";
59
60      cin >> y;
61
62      // le programme verifie le coup
63
64      if (ok=verifier(x,y)) c = maPartie->monDamier->mesCases[x-1][y-1];
65
66    } while (!ok);
67
68    return c;
69
70 }
71
72
73
74 /*void DiableHumain::effectuerCoupSurCase(Case * c) {
75    maPartie->monDamier->mesCases[c->getX()][c->getY()]->setBouchee(true);
76 }*/
77
78
79
80 bool DiableHumain::verifier(int x,int y){
81
82    bool ok=true;
83
84    Case * c;
85
86    if ((x>0) && (y>0) &&
87
88      (x<=maPartie->monDamier->taille)&&
89
90      (y<=maPartie->monDamier->taille)) {
91
92        ok = true;
93
94        c = maPartie->monDamier->mesCases[x-1][y-1];
95
96        if (c->estBouchee()){
97
98          cout << "Erreur: case bouchee." << endl; ok = false;
99
100       }
101
102       if (c->estAnge()) {
103
104         cout << "Erreur: case occupee par l'ange." << endl; ok = false;
105
```

```
106       }
107
108     }
109
110     else ok = false;
111
112   return ok;
113
114 }
```

Listing 13 – Fichier en-tête de la classe DiableAleatoire

```
1  // diableAleatoire.h
2
3
4  #ifndef DIABLE_ALEATOIRE_H
5  #define DIABLE_ALEATOIRE_H
6
7  #include "alea.h"
8  #include "diable.h"
9
10 class Partie;
11
12
13
14 class DiableAleatoire : public Diable {
15 public:
16   DiableAleatoire(Partie *);
17   /*void jouer();*/
18   Case * choisirUneCase();
19   /*void effectuerCoupSurCase(Case *);*/
20 };
21
22 #endif
```

Listing 14 – Classe DiableAleatoire

```
1  // diableAleatoire.cpp
2
3
4
5  #include "diableAleatoire.h"
6
7  #include "partie.h"
8
9
10
11
12
13 DiableAleatoire::DiableAleatoire(Partie * p) :
```

```
14
15 Diable(p)
16
17 {
18
19 }
20
21
22
23 /*void DiableAleatoire::jouer(){
24   cout << "Joueur::jouer(): debut:" << endl;
25   Case * c = choisirUneCase();
26   effectuerCoupSurCase(c);
27   cout << "Joueur::jouer(): fin." << endl;
28 }*/
29
30
31
32
33
34 Case * DiableAleatoire::choisirUneCase() {
35
36     int x = 0;
37
38   int y = 0;
39
40   int t = maPartie->monDamier->taille;
41
42   int i, j, n, r;
43
44   Case * c;
45
46   // le programme choisit un coup
47
48   n = 0; // on compte le nombre de coups possibles.
49
50   for (i=0; i<t; i++) {
51
52     for (j=0; j<t; j++) {
53
54       c = maPartie->monDamier->mesCases[i][j];
55
56       if (!(c->estBouchee()) && (!c->estAnge())) n++;
57
58     }
59
60   }
61
62   r = Alea::engendrer(n);
```

```
63
64    n = 0; // on selectionne un coup aleatoire dans les coups possibles.
65
66    for (i=0; i<t; i++) {
67
68      for (j=0; j<t; j++) {
69
70        c = maPartie->monDamier->mesCases[i][j];
71
72        if (!(c->estBouchee()) && (!c->estAnge()))
73
74          if (++n == r) {
75
76            x = c->getX() + 1;
77
78            y = c->getY() + 1;
79
80          }
81
82        }
83
84    }
85
86    return c;
87
88 }
89
90
91
92 /*void DiableAleatoire::effectuerCoupSurCase(Case * c) {
93    maPartie->monDamier->mesCases[c->getX()][c->getY()]->setBouchee(true);
94 }*/
```

Listing 15 – Fichier en-tête de la classe Partie

```
1  // partie.h
2
3  #ifndef PARTIE_H
4  #define PARTIE_H
5
6  #include "damier.h"
7  #include "joueur.h"
8  #include "case.h"
9  #include "diable.h"
10 #include "ange.h"
11
12 #include "angeHumain.h"
13 #include "angeAleatoire.h"
14 #include "diableHumain.h"
15 #include "diableAleatoire.h"
```

```
16
17 #include <iostream>
18
19 using namespace std;
20
21 class Ange;
22 class Diable;
23 class Damier;
24 class Joueur;
25
26 class Partie {
27 public:
28   Ange * monAnge;
29     Diable * monDiable;
30     Damier * monDamier;
31     Joueur * trait;
32     bool gagnee;
33
34     Partie(int);
35     Partie(int, char, char);
36     bool faire();
37     void initialiser();
38     void afficher();
39     Joueur * autreJoueur();
40 };
41
42 #endif
```

Listing 16 – Classe Partie

```
1 // partie.cpp
2
3 #include "partie.h"
4
5 Partie::Partie(int t, char a, char d) {
6   monDamier = new Damier(t);
7   gagnee = false;
8
9   if (a=='h') monAnge = new AngeHumain(this, 1);
10  if (a=='a') monAnge = new AngeAleatoire(this, 1);
11  Case * c = monDamier->mesCases[monDamier->taille/2][monDamier->taille/2];
12  c->setAnge(monAnge);
13  monAnge->setCase(c);
14  if (d=='h') monDiable = new DiableHumain(this);
15  if (d=='a') monDiable = new DiableAleatoire(this);
16
17  trait = monAnge;
18 }
19
20 Partie::Partie(int t) {
```

```
21   monDamier = new Damier(t);
22   gagnee = false;
23 }
24
25
26 bool Partie::faire() {
27   cout << "Debut de la partie." << endl;
28   monDamier->afficher();
29   do {
30     trait->jouer();
31     monDamier->afficher();
32     gagnee = monAnge->jeSuisBloque() || monAnge->jeSuisLibre();
33     trait = autreJoueur();
34   } while (!gagnee);
35   cout << "Fin de la partie." << endl;
36   return (monAnge->jeSuisBloque());
37 }
38
39 Joueur * Partie::autreJoueur() {
40   if (trait == monAnge) return monDiable;
41   else return monAnge;
42 }
43
44 void Partie::initialiser() {
45   cout << "\tAnge Humain ou Aleatoire ? (h/a)" << endl;
46   char r;
47   cin >> r;
48   if (r=='h') monAnge = new AngeHumain(this, 1);
49   if (r=='a') monAnge = new AngeAleatoire(this, 1);
50   Case * c = monDamier->mesCases[monDamier->taille/2][monDamier->taille/2];
51   c->setAnge(monAnge);
52   monAnge->setCase(c);
53   cout << "\tDiable Humain ou Aleatoire ? (h/a)" << endl;
54   cin >> r;
55   if (r=='h') monDiable = new DiableHumain(this);
56   if (r=='a') monDiable = new DiableAleatoire(this);
57   trait = monAnge;
58 }
59
60 void Partie::afficher() {
61     cout << " mon Ange = ";
62     if (monAnge) cout << monAnge << endl;
63     else cout << " null..." << endl;
64
65     cout << " mon Diable = ";
66     if (monDiable) cout << monDiable << endl;
67     else cout << " null..." << endl;
68
69     cout << " mon Damier = ";
```

```
70      if (monDamier) cout << monDamier << endl;
71      else cout << " null..." << endl;
72
73      cout << " trait = ";
74      if (trait==monAnge)    cout << " >a< " << endl;
75      if (trait==monDiable) cout << "  @  " << endl;
76      if (trait==NULL) cout << " null " << endl;
77
78      cout << " gagnee = ";
79      if (gagnee)    cout << "true " << endl;
80      else cout << "false " << endl;
81  }
```

Listing 17 – Fichier en-tête de la classe Damier

```
1  // damier.h
2
3  #ifndef DAMIER_H
4  #define DAMIER_H
5
6  #include <iostream>
7  #include "case.h"
8
9  using namespace std;
10
11 #define TAILLEMAX 101
12
13 class Case;
14
15 class Damier {
16
17 public:
18   Case * mesCases[TAILLEMAX][TAILLEMAX];
19   int taille;
20   Damier(int);
21   void afficher();
22 };
23
24
25 #endif
```

Listing 18 – Classe Damier

```
1  // damier.cpp
2
3  #include "damier.h"
4
5
6
7  Damier::Damier(int t) {
```

```
8   taille = t;
9   int i, j;
10  for (i=0; i<taille; i++) {
11    for (j=0; j<taille; j++) {
12      mesCases[i][j] = new Case(i, j, this);
13    }
14  }
15 }
16
17
18
19 void Damier::afficher() {
20
21  int i, j;
22  cout << "   ";
23  for (i=1; i<=taille; i++) cout << " " << i << " ";
24  cout << endl;
25  for (i=0; i<taille; i++) {
26    cout << " " << (i+1) << " ";
27    for (j=0; j<taille; j++) {
28      mesCases[i][j]->afficher();
29    }
30    cout << " " << (i+1) << " ";
31    cout << endl;
32  }
33  cout << "   ";
34  for (i=1; i<=taille; i++) cout << " " << i << " ";
35  cout << endl;
36
37 }
```

Listing 19 – Fichier en-tête de la classe Case

```
1  // case.h
2
3  #ifndef CASE_H
4  #define CASE_H
5
6  #include "damier.h"
7  #include "ange.h"
8
9  class Damier;
10 class Ange;
11
12 class Case {
13 private :
14     Damier * monDamier;
15     bool bouchee;
16     Ange * monAnge;
17     int x;
```

```
18      int y;
19  public:
20      Case(int, int, Damier *);
21    int getX(){return x;}
22    int getY(){return y;}
23    bool estBouchee(){return bouchee;}
24    bool estAnge(){return monAnge!=NULL;}
25    void setBouchee(bool b){bouchee=b;}
26    void setAnge(Ange* ma){monAnge=ma;}
27      int distance(Case *);
28      void afficher();
29  };
30
31
32  #endif
```

Listing 20 – Classe Case

```
1  // case.cpp
2
3  #include "case.h"
4
5
6  Case::Case(int a, int b, Damier * d) {
7    monDamier = d;
8    x = a;
9    y = b;
10   monAnge = NULL;
11   bouchee = false;
12 }
13
14
15 int Case::distance(Case * c) {
16   int dx = 0;
17   if (x>=c->x) dx = x - c->x;
18   else dx = c->x - x;
19   int dy = 0;
20   if (y>=c->y) dy = y - c->y;
21   else dy = c->y - y;
22   if (dx>dy) return dx;
23   else return dy;
24 }
25
26
27 void Case::afficher() {
28   if (monAnge != NULL) cout << ">A<";
29   else if (bouchee) cout <<  " @ ";
30   else cout << " + ";
31 }
```

Listing 21 – Fichier en-tête de la classe Alea

```
1 // alea.h
2
3 #ifndef ALEA_H
4 #define ALEA_H
5
6 class Alea {
7 public:
8   static int engendrer(int);
9 };
10
11 #endif
```

Listing 22 – Classe Alea

```
1 // alea.cpp
2
3
4
5 #include "alea.h"
6
7
8
9 #include <iostream>
10
11 #include <time.h>
12
13 #include <math.h>
14
15
16
17 // NOTE : DEBUT CODE pour palier au manque de gettimeofday
18
19 // Copi du net
20
21 #include <windows.h>
22
23
24
25
26
27 #if defined(_MSC_VER) || defined(_MSC_EXTENSIONS)
28
29   #define DELTA_EPOCH_IN_MICROSECS  11644473600000000Ui64
30
31 #else
32
33   #define DELTA_EPOCH_IN_MICROSECS  11644473600000000ULL
34
```

```
35 #endif
36
37
38
39 struct timezone
40
41 {
42
43   int  tz_minuteswest; /* minutes W of Greenwich */
44
45   int  tz_dsttime;     /* type of dst correction */
46
47 };
48
49
50
51 int gettimeofday(struct timeval *tv, struct timezone *tz)
52
53 {
54
55   FILETIME ft;
56
57   unsigned __int64 tmpres = 0;
58
59   static int tzflag;
60
61
62
63   if (NULL != tv)
64
65   {
66
67     GetSystemTimeAsFileTime(&ft);
68
69
70
71     tmpres |= ft.dwHighDateTime;
72
73     tmpres <<= 32;
74
75     tmpres |= ft.dwLowDateTime;
76
77
78
79     /*converting file time to unix epoch*/
80
81     tmpres -= DELTA_EPOCH_IN_MICROSECS;
82
83     tmpres /= 10;  /*convert into microseconds*/
```

```
84
85      tv->tv_sec = (long)(tmpres / 1000000UL);
86
87      tv->tv_usec = (long)(tmpres % 1000000UL);
88
89    }
90
91
92
93    if (NULL != tz)
94
95    {
96
97      if (!tzflag)
98
99      {
100
101       _tzset();
102
103       tzflag++;
104
105     }
106
107     tz->tz_minuteswest = _timezone / 60;
108
109     tz->tz_dsttime = _daylight;
110
111   }
112
113
114
115   return 0;
116
117 }
118
119
120
121 // NOTE : FIN CODE pour palier au manque de gettimeofday
122
123
124
125 int hasard(int n)
126
127 {
128
129   struct timeval tp;
130
131   struct timezone tzp;
132
```

```
133   double x;
134
135   int q, y;
136
137   long r;
138
139
140
141   gettimeofday(&tp,&tzp);
142
143   r = tp.tv_usec;
144
145   x = (double) r/n;
146
147   q = (int) floor(x);
148
149   y = (int) r - n*q + 1;
150
151   //cout << "Nombre au hasard entre 1 et " << n << ": " << d << "\n";
152
153   return y;
154
155 }
156
157
158
159
160
161 int Alea::engendrer(int n) {
162
163   int a = hasard(n);
164
165   // cout << "n = " << n << " a = " << a << endl;
166
167   return a;
168
169 }
```

Listing 23 – Main

```
1  // main.cpp
2
3  #include <iostream>
4  #include "partie.h"
5
6  using namespace std;
7
8  void main () {
9    cout << "Jeu de l'Ange et du Diable." << endl << endl;
10   char r;
```

```
11    int t;
12    Partie * p = NULL;
13
14    do {
15      cout << "quitter              (q)" << endl;
16      cout << "faire une partie     (f)" << endl;
17      cin >> r;
18      if (r=='f') {
19        cout << "\tTaille du damier ? " << endl;
20        cin >> t;
21        p = new Partie(t);
22        p->initialiser();
23        cout << "Voici la partie: " << endl;
24        p->afficher();
25        cout << "Voici l'ange: " << endl;
26        p->monAnge->afficher();
27        cout << "Voici le diable: " << endl;
28        p->monDiable->afficher();
29        cout << "Voici faire: " << endl;
30        cout << p->faire() << endl;
31      }
32    } while (r!='q');
33
34
35    cout << "Au revoir." << endl;
36
37
38  }
```