

Projet Bataille navale en C

Conception du projet

► Thème

La bataille navale, appelée aussi touché-coulé, est un jeu de société dans lequel deux joueurs doivent placer des « navires » sur une grille tenue secrète et tenter de « toucher » les navires adverses.

Conception du projet

► Objectif du joueur

- Le joueur doit découvrir la position des navires ennemis et les couler. L'objectif ultime est de couler tous les navires adverses avant que l'adversaire ne coule les siens.
- Observation et mémoire (se rappeler où on a tiré, observer les schémas de tir de l'adversaire).
- Stratégie (placer ses navires de manière à maximiser leurs chances de survie et tirer sur des zones probables de l'adversaire).

Conception du projet

► Mécaniques de jeu principales

Placement des navires

Le placement des navires sur une grille de taille 8x8 est une phase clé, car il influence la partie entière.

Les joueurs choisissent une coordonnée sur la grille adverse pour tenter de toucher un navire cela se base sur la chance tour par tour

La simplicité accessible des mécaniques qui laisse place à une grande profondeur stratégique au fil des parties.

Conception du projet

- Mécaniques de jeu principales

Sauvegarde de la partie :

Les deux joueurs peuvent sauvegarder la partie à tout moment

Présentation du groupe

- ▶ Kilian Jeny (Développeur)
- ▶ Baptiste Paris (Développeur)
- ▶ Samy Boumahrat (Chef de groupe)
- ▶ Hamza Kassou (Présentation Powerpoint)

Trello

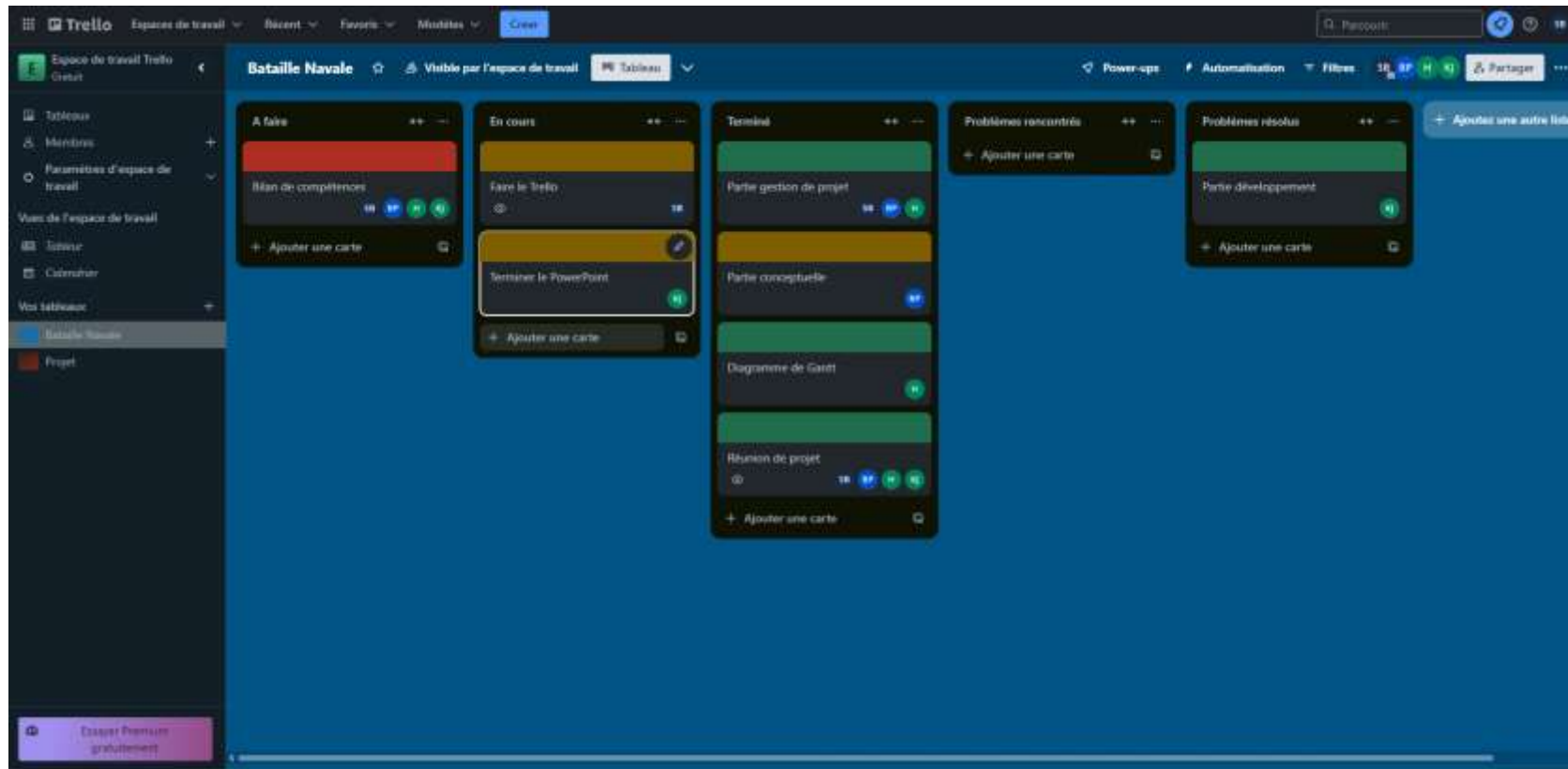


Diagramme de Pert

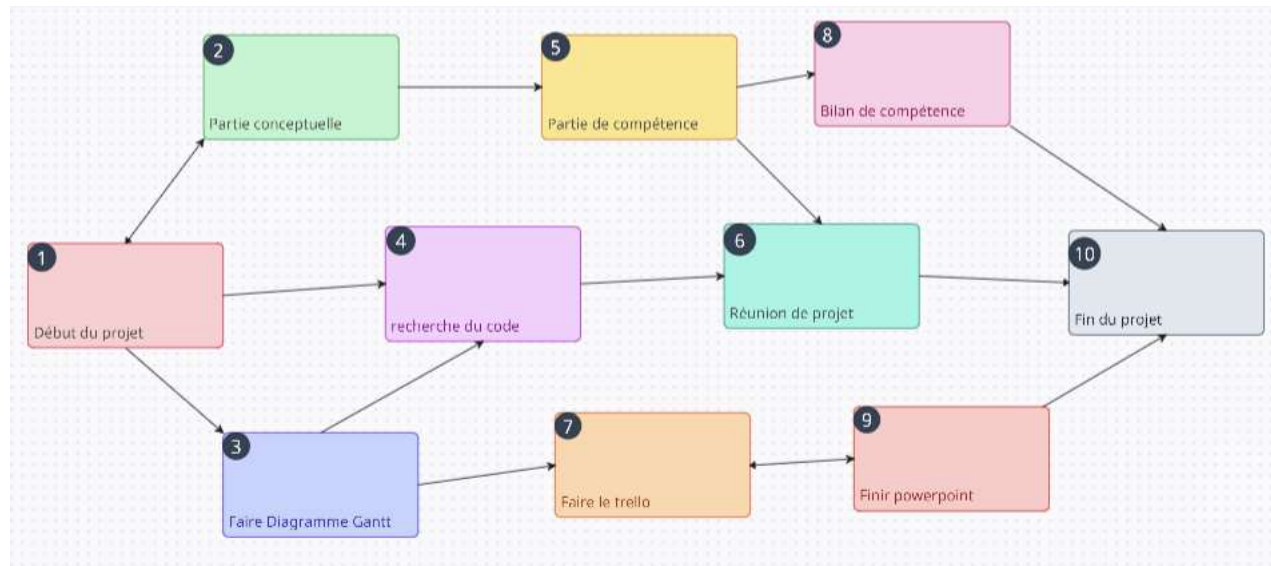


Diagramme de Gantt

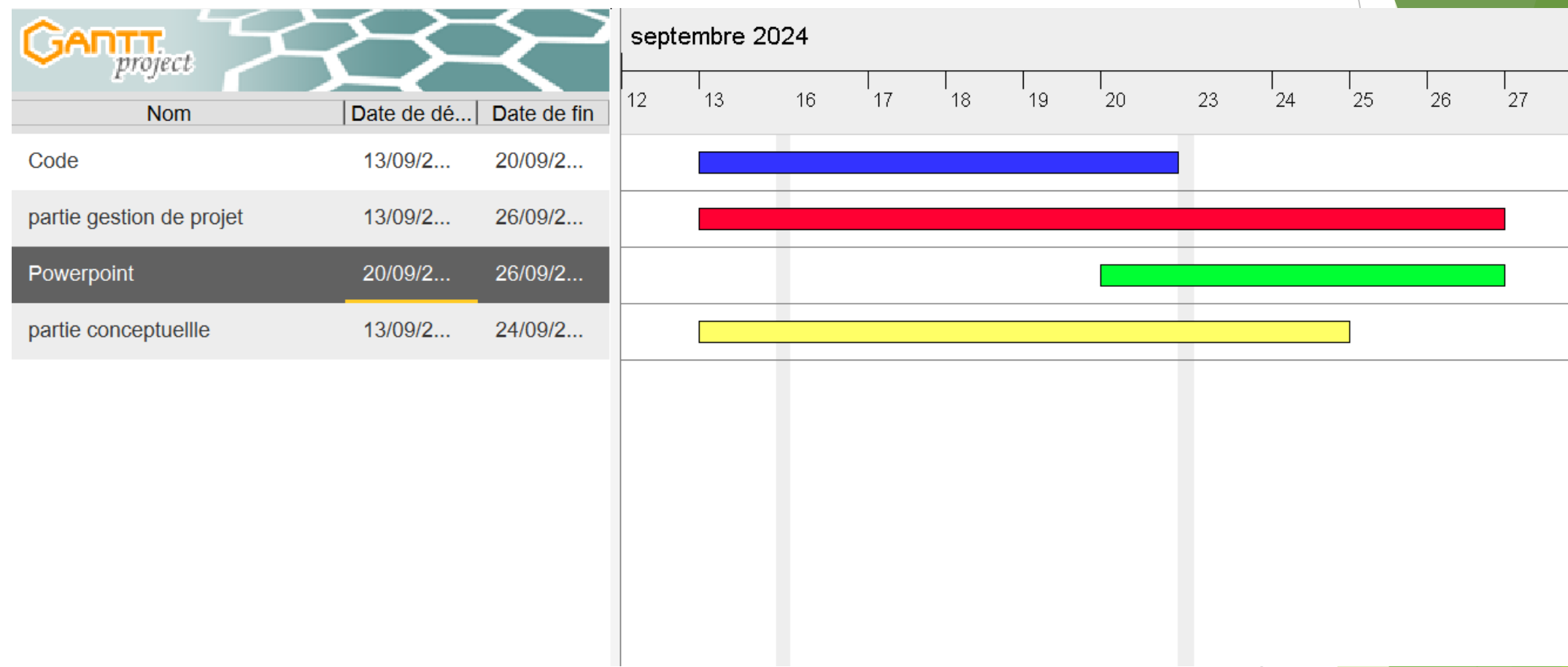
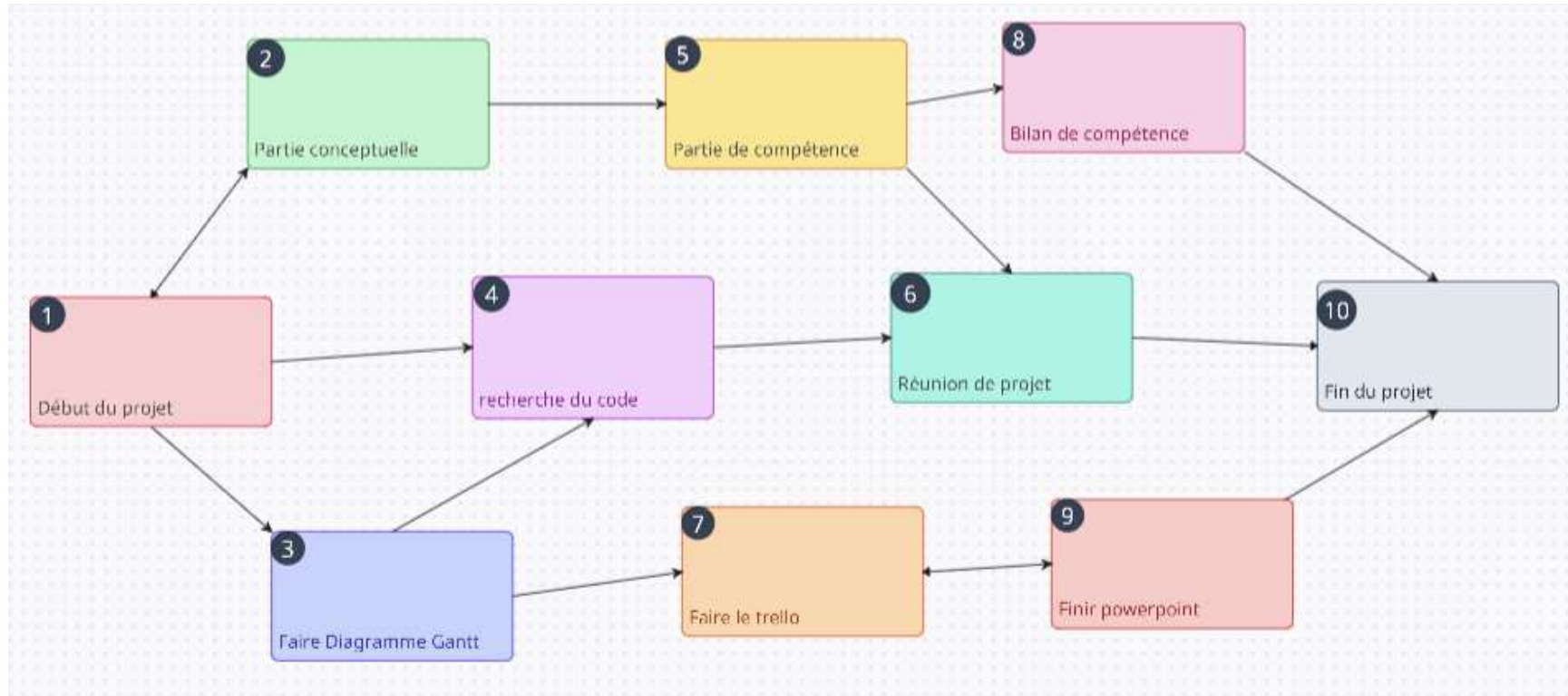


Diagramme de Pert



1. Les bibliothèques

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <time.h>
```

2. Définitions des constantes

- ▶ `#define TAILLE_GRILLE 8`
- ▶ `#define NB_BATEAUX 3`
- ▶ `#define JOUEUR_1 1`
- ▶ `#define JOUEUR_2 2`

3. Définir la grille

```
► typedef struct {  
►     int grille[TAILLE_GRILLE][TAILLE_GRILLE];  
► } Grille;  
  
► void initialiserGrille(Grille *grille) {  
►     for (int i = 0; i < TAILLE_GRILLE; i++) {  
►         for (int j = 0; j < TAILLE_GRILLE; j++) {  
►             grille->grille[i][j] = 0;  
►         }  
►     }  
► }
```

3. Définir la grille

```
▶ void afficherGrille(Grille *grille) {  
▶     printf(" ");  
▶     for (int i = 0; i < TAILLE_GRILLE; i++) {  
▶         printf("%d ", i);  
▶     }  
▶     printf("\n");  
▶     for (int i = 0; i < TAILLE_GRILLE; i++) {  
▶         printf("%d ", i);
```

3. Définir la grille

```
▶ for (int j = 0; j < TAILLE_GRILLE; j++) {  
▶     if (grille->grille[i][j] == 0)  
▶         printf("~ ");  
▶     else if (grille->grille[i][j] == -1)  
▶         printf("X ");  
▶     else  
▶         printf("B ");  
▶ }  
▶ printf("\n");  
▶ }  
▶ }
```

4. Définir les bateaux

- ▶ `int placerBateauManuellement(Grille *grille, int taille) {`
- ▶ `int x, y, orientation;`
- ▶ `printf("Placer un bateau de taille %d\n", taille);`
- ▶ `printf("Entrez les coordonnées de départ (x y) : ");`
- ▶ `scanf("%d %d", &x, &y);`
- ▶ `printf("Choisissez l'orientation (0 pour horizontal, 1 pour vertical) : ");`
- ▶ `scanf("%d", &orientation);`

4. Définir les bateaux

```
▶ if (orientation == 0) { // Horizontal
▶     if (y + taille > TAILLE_GRILLE) {
▶         printf("Le bateau dépasse la grille. Choisissez un autre emplacement.\n");
▶         return 0; // Échec
▶     }
▶     for (int i = 0; i < taille; i++) {
▶         if (grille->grille[x][y + i] != 0) {
▶             printf("Le bateau chevauche un autre bateau. Choisissez un autre emplacement.\n");
▶             return 0; // Échec
▶         }
▶     }
▶     for (int i = 0; i < taille; i++) {
▶         grille->grille[x][y + i] = taille;
▶     }
```

4. Définir les bateaux

```
▶ } else { // Vertical
▶     if (x + taille > TAILLE_GRILLE) {
▶         printf("Le bateau dépasse la grille. Choisissez un autre emplacement.\n");
▶         return 0; // Échec
▶     }
▶     for (int i = 0; i < taille; i++) {
▶         if (grille->grille[x + i][y] != 0) {
▶             printf("Le bateau chevauche un autre bateau. Choisissez un autre
emplacement.\n");
▶             return 0; // Échec
▶         }
▶     }
▶     for (int i = 0; i < taille; i++) {
▶         grille->grille[x + i][y] = taille;
▶     }
▶ }
▶ return 1; // Succès
▶ }
```

4. Définir les bateaux

```
▶ void placerBateaux(Grille *grille) {  
▶     int tailles_bateaux[NB_BATEAUX] = {2, 3, 4};  
▶     for (int i = 0; i < NB_BATEAUX; i++) {  
▶         int place = 0;  
▶         while (!place) {  
▶             place = placerBateauManuellement(grille, tailles_bateaux[i]);  
▶         }  
▶     }  
▶ }
```

5. Définir le tir

```
► int tirer(Grille *grille, int x, int y) {  
►     if (grille->grille[x][y] > 0) {  
►         int tailleBateau = grille->grille[x][y];  
►  
►         // Parcourir la grille et marquer tout le bateau comme coulé  
►         for (int i = 0; i < TAILLE_GRILLE; i++) {  
►             for (int j = 0; j < TAILLE_GRILLE; j++) {  
►                 if (grille->grille[i][j] == tailleBateau) {  
►                     grille->grille[i][j] = -1; // Marquer la case comme coulé  
►                 }  
►             }  
►         }  
►         return 1; // Le bateau a été touché et coulé  
►     }  
►     return 0; // Manqué  
► }
```

6. Définir l'état « coulé »

```
► int estCoule(Grille *grille, int taille) {  
►     // Vérifie si tous les segments du bateau de taille donnée sont touchés  
►     for (int i = 0; i < TAILLE_GRILLE; i++) {  
►         for (int j = 0; j < TAILLE_GRILLE; j++) {  
►             if (grille->grille[i][j] == taille) {  
►                 return 0; // Le bateau n'est pas complètement coulé  
►             }  
►         }  
►     }  
►     return 1; // Le bateau est coulé  
► }
```

7. Définir la victoire

```
▶ int estGagne(Grille *grille) {  
▶     for (int i = 0; i < TAILLE_GRILLE; i++) {  
▶         for (int j = 0; j < TAILLE_GRILLE; j++) {  
▶             if (grille->grille[i][j] > 0) {  
▶                 // S'il y a encore une case qui n'est pas coulé (> 0), la partie continue  
▶                 return 0;  
▶             }  
▶         }  
▶     }  
▶     // Si toutes les cases de bateau sont coulées, le joueur a perdu  
▶     return 1;  
▶ }
```

8. Sauvegarder la partie

- ▶ `void sauvegarderPartie(Grille *grilleJ1, Grille *grilleJ2, const char *nomFichier) {`
- ▶ `FILE *fichier = fopen(nomFichier, "wb");`
- ▶ `if (fichier == NULL) {`
- ▶ `printf("Erreur lors de la sauvegarde de la partie.\n");`
- ▶ `return;`
- ▶ `}`
- ▶ `fwrite(grilleJ1, sizeof(Grille), 1, fichier);`
- ▶ `fwrite(grilleJ2, sizeof(Grille), 1, fichier);`
- ▶ `fclose(fichier);`
- ▶ `printf("Partie sauvegardée.\n");`

9. Charger la partie

```
▶ void chargerPartie(Grille *grilleJ1, Grille *grilleJ2, const char *nomFichier) {  
▶     FILE *fichier = fopen(nomFichier, "rb");  
▶     if (fichier == NULL) {  
▶         printf("Erreur lors du chargement de la partie.\n");  
▶         return;  
▶     }  
▶     fread(grilleJ1, sizeof(Grille), 1, fichier);  
▶     fread(grilleJ2, sizeof(Grille), 1, fichier);  
▶     fclose(fichier);  
▶     printf("Partie chargée.\n");  
▶ }
```


10. Le choix entre tirer et sauvegarder

- ▶ `void jouerTour(Grille *grilleAdverse, int joueur, Grille *grilleJ1, Grille *grilleJ2) {`
- ▶ `int choix;`
- ▶ `int x, y;`
- ▶ `while (1) {`
- ▶ `printf("Joueur %d, que voulez-vous faire ?\n", joueur);`
- ▶ `printf("1. Tirer\n2. Sauvegarder la partie\nChoix : ");`
- ▶ `scanf("%d", &choix);`

10. Le choix entre tirer et sauvegarder

```
▶ if (choix == 1) {  
▶     // Tirer  
▶     printf("Entrez les coordonnées (x y) de votre tir : ");  
▶     scanf("%d %d", &x, &y);  
▶     if (x < 0 || x >= TAILLE_GRILLE || y < 0 || y >= TAILLE_GRILLE) {  
▶         printf("Coordonnées invalides. Réessayez.\n");  
▶     } else {  
▶         if (tirer(grilleAdverse, x, y)) {  
▶             printf("Touché et coulé !\n");  
▶         } else {  
▶             printf("Manqué.\n");  
▶         }  
▶         break; // Fin du tour  
▶     }  
▶ }
```

10. Le choix entre tirer et sauvegarder

```
▶ } else if (choix == 2) {  
▶     // Sauvegarder la partie  
▶     sauvegarderPartie(grilleJ1, grilleJ2, "sauvegarde.dat");  
▶     printf("La partie a été sauvegardée.\n");  
▶ } else {  
▶     printf("Choix invalide. Réessayez.\n");  
▶ }  
▶ }  
▶ }
```

11. Déroulement de la partie

```
▶ void jeu(Grille *grilleJ1, Grille *grilleJ2) {  
▶     int tour = JOUEUR_1;  
  
▶     while (1) {  
▶         printf("\nGrille du joueur %d :\n", JOUEUR_1);  
▶         afficherGrille(grilleJ1);  
▶         printf("\nGrille du joueur %d :\n", JOUEUR_2);  
▶         afficherGrille(grilleJ2);  
  
▶         if (tour == JOUEUR_1) {  
▶             jouerTour(grilleJ2, JOUEUR_1, grilleJ1, grilleJ2);  
▶             if (estGagne(grilleJ2)) {
```

11. Déroulement de la partie

```
▶ printf("Joueur 1 a gagné !\n");
▶     break; // Fin du jeu
▶ }
▶ tour = JOUEUR_2;
▶ } else {
▶     jouerTour(grilleJ1, JOUEUR_2, grilleJ1, grilleJ2);
▶     if (estGagne(grilleJ1)) {
▶         printf("Joueur 2 a gagné !\n");
▶         break; // Fin du jeu
▶     }
▶     tour = JOUEUR_1;
▶ }
▶ }
▶ }
```

12. Le main

```
▶ int main() {  
▶     srand(time(NULL));  
  
▶     int choix;  
▶     Grille grilleJ1, grilleJ2;  
▶  
▶     printf("1. Nouvelle partie\n2. Charger une partie\nChoix : ");  
▶     scanf("%d", &choix);  
  
▶     if (choix == 2) {  
▶         chargerPartie(&grilleJ1, &grilleJ2, "sauvegarde.dat");  
▶     }
```

12. Le main

```
▶ } else {  
▶     initialiserGrille(&grilleJ1);  
▶     initialiserGrille(&grilleJ2);  
  
▶     printf("Joueur 1, placez vos bateaux.\n");  
▶     placerBateaux(&grilleJ1);  
▶     printf("Joueur 2, placez vos bateaux.\n");  
▶     placerBateaux(&grilleJ2);  
  
▶     printf("\nLes deux joueurs ont placé leurs bateaux. Début de la partie !\n");  
▶ }
```

12. Le main

```
▶ // Boucle principale du jeu
▶ jeu(&grilleJ1, &grilleJ2);

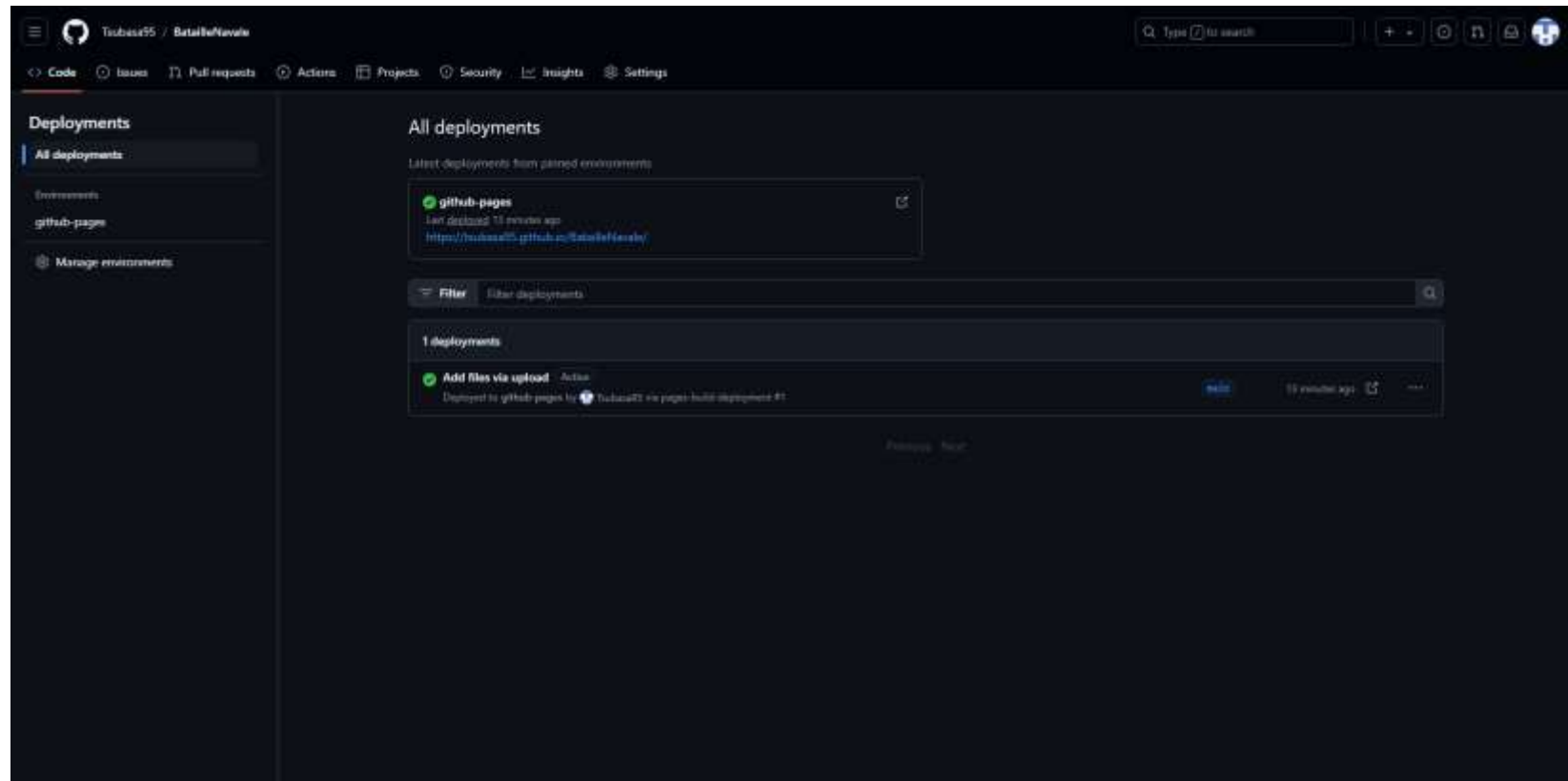
▶ // À la fin de la partie, possibilité de sauvegarder
▶ printf("Voulez-vous sauvegarder la partie (1 pour oui, 0 pour non) ? ");
▶ scanf("%d", &choix);
▶ if (choix == 1) {
▶     sauvegarderPartie(&grilleJ1, &grilleJ2, "sauvegarde.dat");
▶ }

▶ return 0;
▶ }
```

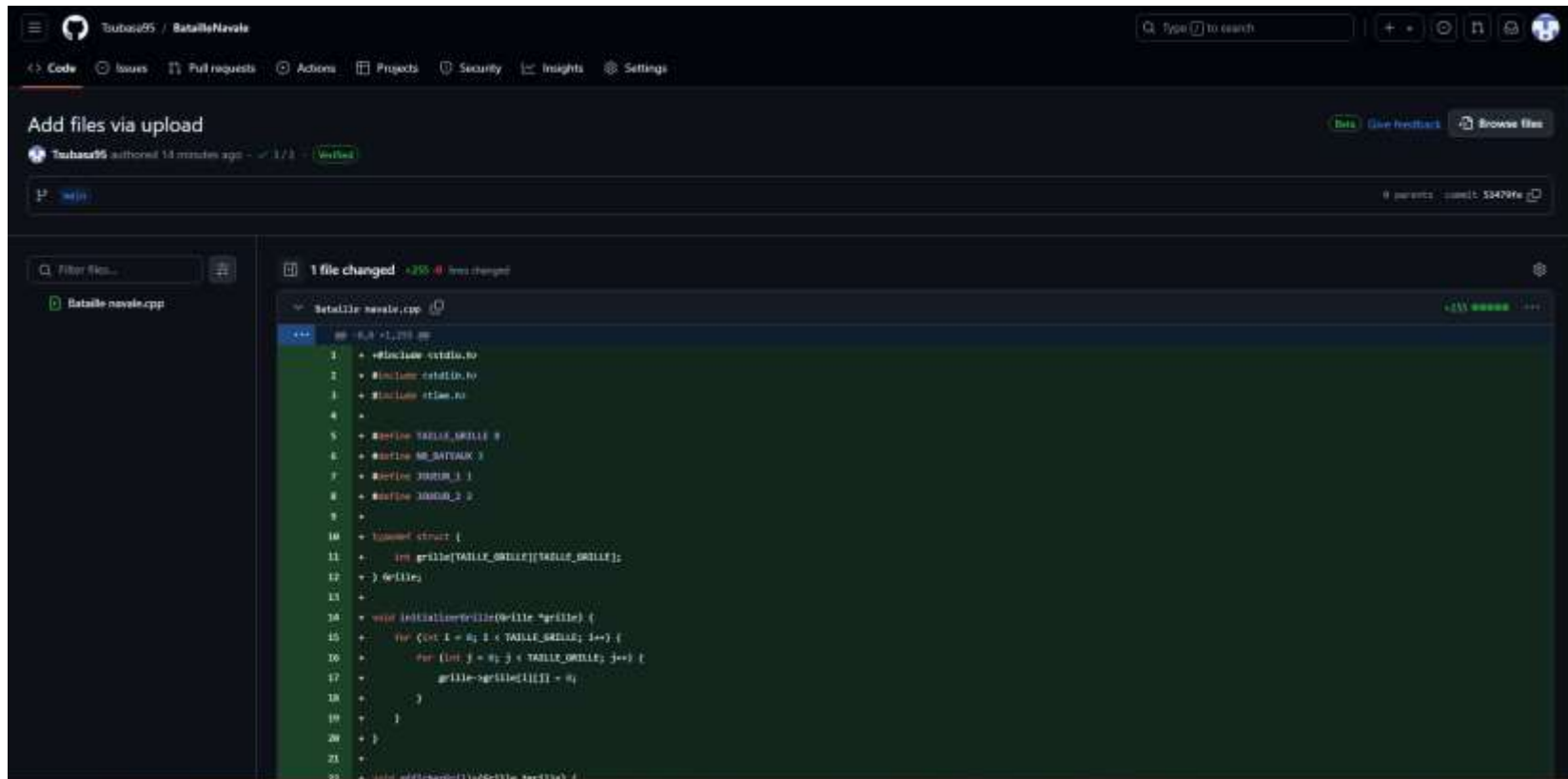

Le bilan de compétence

- ▶ Meilleure maîtrise en C
- ▶ Sauvegarder et charger un fichier en C
- ▶ Utilisation de matrice

Github



Github



The screenshot shows a GitHub repository interface for 'Toubac95 / BatailleNavale'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. A search bar is located on the right. The main content area is titled 'Add files via upload' and shows a commit by 'Toubac95' from 13 minutes ago. Below this, a file named 'BatailleNavale.cpp' is listed. The diff view for this file shows changes to 'BatailleNavale.cpp', with a summary indicating '1 file changed' and '+255 -0 lines changed'. The code is displayed in a dark-themed editor with line numbers on the left. The code includes standard C++ headers and defines a grid-based game logic.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #define TABLE_SIZE 8
6  #define NB_SHIPS 3
7  #define SHIP_SIZE 1
8  #define SHIP_SIZE 2
9
10 #typedef struct {
11     int grille[TABLE_SIZE][TABLE_SIZE];
12 } Grille;
13
14 void InitialiserGrille(Grille *g) {
15     for (int i = 0; i < TABLE_SIZE; i++) {
16         for (int j = 0; j < TABLE_SIZE; j++) {
17             grille->grille[i][j] = 0;
18         }
19     }
20 }
21
22 void AfficherGrille(Grille *g) {
23     printf("Grille du joueur 1:\n");
24     for (int i = 0; i < TABLE_SIZE; i++) {
25         for (int j = 0; j < TABLE_SIZE; j++) {
26             printf("%d ", grille->grille[i][j]);
27             if (j % 10 == 9) printf("\n");
28         }
29     }
30 }
```