RAPPORT : ORGANISATION

Organisation adoptée par l'équipe :

Sprint A :
Jérémie Sold : Référent, responsable de l'écriture de la documentation, de l'organisation
Min Li : Chargée de développer les fichiers grille.c et grille.h
Yoann Gouet : Chargé de développer les fichiers couleur.c et couleur.h
Aymeric Fraise : Chargé de développer les tests unitaires

Sprint B :
Aymeric Fraise : Référent, documention, rapports
Min Li : Développement du jeu, SDL
Jérémie Sold : Refactoring
Yoann Gouet : Développement du jeu, documentation

RAPPORT : ALGORITHMES

GRILLE.C
1. On définit une structure Case qui est composée d'une variable couleur et d'une variable "dedans". Cette dernière permettra de déterminer par la suite si la case en question sera comprise dans la tâche de la case supérieur gauche.

```
struct Case{
  char couleur;
  int dedans;
};
```

2. La grille est composée de cases. On écrit une fonction pour créer une grille de taille carrée. On utilise la fonction malloc pour obtenir un espace continu. Taille représente la longueur d'un côté.

```
grille Grille(int taille)
{
      grille ret=NULL;
      ret=malloc(taille*taille*sizeof(Case));
      return ret;
}
```

3. Pour libérer l'espace mémoire occupé par une grille, on a écrit une fonction qui s'appelle Liberation(grille a) en utilisant la fonction free();

```
void Liberation(grille a)
{
   if(a!=NULL)
  {
   free(a);
   a=NULL;
  }
}
```

4.pour avoir une couleur aléatoire,on utilise la fonction random() pour créer la couleur aléatoire

```
char constructeur()
{
   /*srand((unsigned)time(NULL));*/
```

```
    int i=random()%6+1;
    char couleur;
    switch(i)
   {
     case 1:couleur='B';break;
     case 2:couleur='V';break;
     case 3:couleur='R';break;
     case 4:couleur='J';break;
     case 5:couleur='M';break;
     case 6:couleur='G';break;
     default:break;
   }
    return couleur;
}
```

5.on écrit une fonction qui s'appelle get_couleur(grille g) pour obtenir la couleur d'une case

```
char get_couleur(grille g) {
        return (g->couleur);
}
```

6.on écrit une fonction qui s'appelle get_dedant(grille g)pour obtenir la valeur dedant d'une case

```
int get_dedant(grille g) {
        return (g->dedant);
}
```

7.on écrit une fonction qui s'appelle augemente_pointeur(grille g)pour augmenter la valeur de pointeur g.

8.pour initialiser la grille à partir de valeurs aléatoires,on utilise la fonction ci-dessous.

```
void init_grille(grille a,int taille)
{
    int i=0;
    for(i=0;i<taille*taille;i++)
    {
       a->couleur=constructeur();
       a=augemente_pointeur(a);
    }
}
```

9.pour initialiser la grille à partir de valeurs contenues dans un fichier ,on écrit une fonction

```
void init_grille_fichier(grille a,FILE *fp,int taille)
{
    int i=0;
    for(i=0;i<taille*taille;i++)
    {
       a[i].couleur=fgetc(fp);
    }
    fclose(fp);
}
```

COULEUR.C :

1. On définit d'abord la structure de liste pour stocker la composante connexe.

```
struct t_liste {
  int position;
  struct t_liste *suivant;
};
```

2. listevide est un test de vacuité.

3. makel est le constructeur il place l'entier au sommet de la liste.

4. pop recupere la tete.

5. sui renvoie le suivant.

6. freel libère la mémoire.

7. change1 change la valeur d'une case dans la grille.

Gestion des composantes connexes :
8. La gestion de la composante connexe ce fait en deux instances au niveaux d'une liste et de chaque case.
Chaque case sait si elle appartient à la composante.
A chaque instant la fonction composante met a jour la liste des case appartenant a la composante et met a jour ces dernière.

9. changeall applique change1 a chaque case de la liste.

10. win regarde si toute le case sont dans la composante.

A chaque itération :
-on demande une couleur
-changeall(grille,composante(grille,liste),couleur)
-win(grille,taille)