INFO0030 : Projet de Programmation
Five or More

B. Donnet, K. Edeline, E. Marechal, M. Goffart, E. Wansart

Alerte : Évitez de perdre bêtement des points...

- Nous vous conseillons **vivement** de relire et d'appliquer les **guides de style et de langage**, mis à votre disposition sur eCampus (INFO0030, Sec. Supports pour le Cours Théorique). Cela vous permettra d'éviter de nombreuses erreurs et de perdre des points inutilement.
- Nous vous conseillons de consulter la **grille de cotation** utilisée pour la correction des projets afin d'éviter de perdre bêtement des points. Elle est disponible sur eCampus (INFO0030, Sec. Procédures d'Évaluation).
- Prenez le temps de lire attentivement l'énoncé. La plupart des réponses aux questions que vous vous posez s'y trouvent.
- Votre code sera compilé et testé sur les **machines CANDI qui servent de référence**. La procédure à suivre pour se connecter en SSH aux machines CANDI est disponible sur eCampus. Veillez donc à ce que votre code fonctionne dans cet environnement. Si vous n'avez pas de compte sur ces machines, veuillez contacter Marc Frédéric.
- Votre solution ne pourra pas être générée entièrement ou partiellement à l'aide d'un outil d'Intelligence Artificielle (e.g., ChatGPT, Blackbox, etc.).

1 Contexte

Five or More est un jeu disponible gratuitement en ligne et pour de nombreuses distributions Linux.

Five or More est, en fait, une version Linux d'un célèbre jeu Windows : *Color Lines*. L'objectif du jeu est d'aligner aussi souvent que possible cinq (ou plus) objets de la même couleur et de même forme de façon à les faire disparaître de la surface de jeu. La Figure 1 donne un exemple du jeu.

Dans ce projet, vous devrez, en **binôme**, implémenter une version du Five or More en suivant le pattern **Modèle-Vue-Contrôleur** (MVC) étudié durant le cours théorique.

2 Le Jeu

Cette section décrit le fonctionnement général du jeu Five or More. Nous commençons par présenter les règles du jeu (Sec. 2.1). Ensuite, nous expliquons comment calculer les points tout au long d'une partie (Sec. 2.2). Enfin, nous terminons en discutant les différentes tailles du plateau de jeu (Sec. 2.3).

2.1 Règles

Au début du jeu, il y a 3 ou 7 symboles disposés sur le plateau en fonction de la taille du plateau (cfr. Sec. 2.3).

À chaque fois que le joueur déplace un symbole (avec la souris, en cliquant d'abord sur le symbole "cible" et, ensuite, sur la case "cible", i.e., la case destination du symbole), des symboles supplémentaires sont placés aléatoirement sur le plateau. Le nombre de nouveaux symboles placés à chaque tour dépend de la taille du plateau. Une pré-visualisation des symboles qui vont être placés apparaît dans le coin supérieur gauche de la fenêtre du jeu.

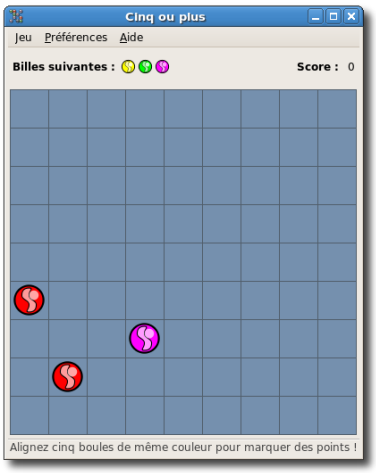


FIGURE 1 – Exemple de FIVE OR MORE.

Le chemin entre la position initiale du symbole et la cellule de destination doit être libre (i.e., vous ne pouvez pas sauter par dessus des symboles).

Une ligne droite horizontale, verticale ou diagonale de cinq symboles similaires ou plus disparaît automatiquement du plateau de jeu dès qu'elle est créée. Lorsque cela se produit, le joueur bénéficie d'un mouvement supplémentaire sans ajout de symboles sur le plateau et son score se met à jour (cfr. Sec. 2.2).

Plus le joueur efface d'objets, plus la partie est longue. Plus il joue longtemps, plus son score augmente.

La partie est terminée lorsque le plateau est plein, puisqu'aucun symbole ne peut plus être déplacé.

Il est **fortement** recommandé de tester une version en ligne du jeu avant de se lancer dans l'implémentation de ce dernier.

2.2 Scores

Le joueur marque des points en créant des lignes (horizontale, verticale ou diagonale) de cinq symboles ou plus. Le nombre de points dépend du nombre de symboles composant la ligne qui disparaît. Le Tableau 1 résume les différents points en fonction de la taille d'une ligne.

# **Symboles**	# **Points**
5	10
6	12
7	18
8	28
9	42
10	82
11	108
12	138
13	172
14	210

TABLE 1 – Scores en fonction du nombre de symboles alignés.

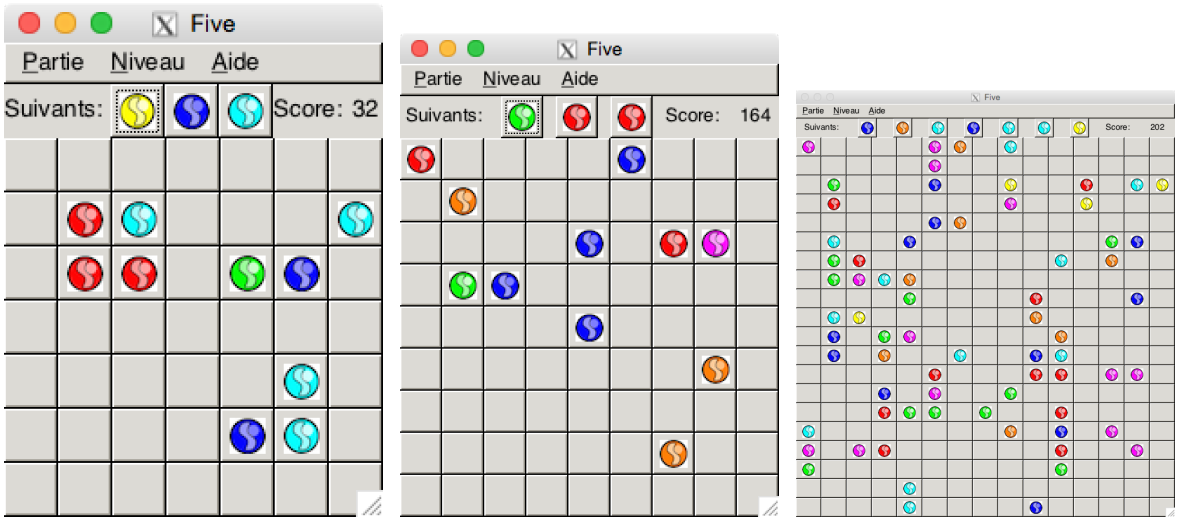
2.3 Taille du Plateau

Le jeu FIVE OR MORE dispose de trois niveaux de difficulté, chacun correspondant à un plateau de jeu de taille différente. Le tableau 2 résume les différents niveaux, avec les dimensions afférentes. Les

différentes tailles du plateau sont illustrées à la Fig. 2.

Difficulté	Largeur	Hauteur	# Types de Symboles	# Symboles/Tour	# Cellules
Facile	7	7	5	3	49
Normal	9	9	7	3	81
Difficile	20	15	7	7	300

TABLE 2 – Résumé des dimensionnements du jeu.



(a) Facile (b) Normal (c) Difficile

FIGURE 2 – Dimensionnement du plateau de jeu en fonction du niveau de difficulté

3 Étapes Importantes

L'exécution du jeu nécessite plusieurs étapes :

1. *Initialisation.* Durant cette étape, les éventuels paramètres en ligne de commande sont lus par le programme et, dans tous les cas, un tirage aléatoire du tableau à deux dimensions a lieu. Ce dernier élément permet de déterminer des emplacements aléatoires pour les premiers symboles, dans le plateau de jeu (cfr. Sec. 4.1).
2. *Interface Graphique.* L'interface graphique (IHM) du jeu est chargée (cfr. Sec. 5).
3. *Déplacement d'un Symbole.* A chaque étape de la partie, le joueur peut déplacer un symbole vers une case libre du plateau de jeu. Le déplacement n'est possible que si il existe un chemin entre l'emplacement actuel du symbole et l'emplacement ciblé, i.e., aucun symbole n'obstrue le chemin (cfr. Sec. 4.2). Le chemin peut être (mais pas obligatoirement) rectiligne.
4. *Alignement de Symboles.* Lorsque le joueur a aligné au moins cinq symboles de même couleur (horizontalement, verticalement, ou en diagonale – la Fig. 3 donne un exemple des alignements possibles pour le symbole #), la ligne de symboles disparaît du plateau de jeu, le joueur se voit créditer les points en conséquences (cfr. Sec. 2.2) et il a le droit d'effectuer encore un déplacement sans voir apparaître de symbole supplémentaire sur le plateau de jeu.
5. *Apparition de Symboles.* Après un déplacement (et si le joueur n'a pas réussi d'alignement), un certain nombre de nouveaux symboles apparaissent sur le plateau (symboles ayant été annoncés par le jeu) en fonction du niveau de difficulté (cfr. Sec. 2.3). Ces symboles doivent être placés, intelligemment, dans des cases libres (cfr. Sec. 4.1).

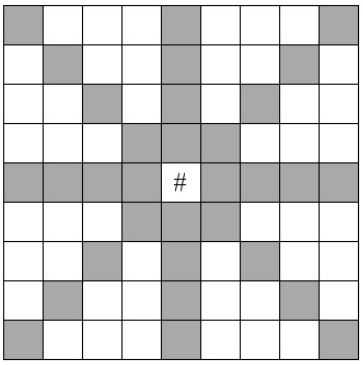


FIGURE 3 – Alignements possibles du symbole #

6. *Fin d'une Partie.* La partie est terminée lorsque le plateau est plein (aucun mouvement n'est possible pour le joueur). Le score final est donné au joueur et si cela s'avère pertinent, son score est sauvé dans la liste des 10 meilleurs scores (cfr. Sec. 5).

4 Algorithmes Importants

Cette section décrit deux algorithmes importants pour le jeu FIVE OR MORE, le placement des symboles sur le plateau de jeu (Sec. 4.1) et le déplacement d'un symbole d'une case à l'autre (Sec. 4.2).

4.1 Placement des Symboles

Pour pouvoir placer des symboles aléatoirement dans le plateau de jeu, il faut utiliser un générateur de nombres aléatoires (cfr. Sec. 6.2). Si le plateau est de dimension $n \times m$, il suffit de tirer aléatoirement un chiffre entre 0 et $n-1$ afin d'obtenir le numéro d'une ligne du plateau. Ensuite, on réitère le tirage aléatoire mais, cette fois, entre 0 et $m-1$, ce qui nous donnera le numéro d'une case sur la ligne choisie.

Il suffit de répéter ce schéma jusqu'à ce qu'on ait placé tous les symboles.

Cet algorithme est assez simple à implémenter mais fortement inefficace dans les cas où le plateau de jeu est plein, voire très plein. Si par exemple on décide de jouer sur une grille de niveau difficile et qu'on a déjà placé 290 symboles, lorsqu'on débutera l'algorithme pour placer les trois suivants, il y a de fortes chances (90%) que la case tirée au hasard soit déjà occupée par un symbole et qu'on doive refaire le tirage au sort de nombreuses fois.

Il est donc souhaitable, dans le cadre de ce projet, de réfléchir à un algorithme plus efficace.

4.2 Calcul du Chemin

Déplacer un symbole d'un endroit à l'autre sans rencontrer d'obstacle(s) s'apparente, d'une certaine façon, à trouver le chemin le plus court entre deux points dans un labyrinthe. Dans cette section, nous vous décrivons comment on peut trouver un chemin entre deux points dans un labyrinthe. [1]

De manière générale, on définit un *labyrinthe* comme étant un tableau de dimensions $N \times M$ dans lequel certaines cases sont occupées. On peut passer d'une case à une des quatre cases voisines si et seulement si cette dernière n'est pas occupée (par un mur – ou dans le cas du FIVE OR MORE, un symbole). Le labyrinthe définit aussi un *point de départ* et un *point d'arrivée*.

L'algorithme employé ici pour résoudre ce problème procède de manière itérative, en recherchant à chaque itération les cases atteignables en $n+1$ pas à partir de celles atteignables en n pas.

Soit C, l'ensemble des cases non occupées. Pour $c \in C$, le couple (c, i) est dans l'ensemble Att (ensemble des cases atteignables) si et seulement si la case c est atteignable en i pas depuis la case de départ. D est la case de départ et A la case d'arrivée. Enfin, $V(c)$ désigne les cases voisines de la case c.

Avec ces notations, l'algorithme 1 permet de calculer le chemin entre D et A.

1. Libre à vous d'implémenter cette méthode ou de réfléchir à votre propre méthode.

Algorithm 1 Calcul du chemin dans un labyrinthe.

```
1:  Att = (D, 0)
2:  step ← 0
3:  while ∀i, (A, i) ∉ Att do
4:      for all c such that (c, step) ∈ Att do
5:          for all c' ∈ V(c) ⋂ C do
6:              if (c', i) ∉ Att then
7:                  Att = Att ⋃ (c', step + 1)
8:              end if
9:          end for
10:         step ← step + 1
11:     end for
12: end while
```

Dans l'implémentation de l'algorithme, on pourra simplement utiliser un tableau dont la valeur de chaque case représente son état (case occupée, case de départ, case d'arrivée, case pas encore atteinte, case atteinte).

Enfin, il est bon de noter que cet algorithme ne fait pas le tracé du chemin. Pour ce faire, il faut partir de l'arrivée et remonter jusqu'à la case de départ en trouvant à chaque itération une case atteignable avec un nombre de pas égal au nombre de pas de la case actuelle moins un.

5 Interface Graphique

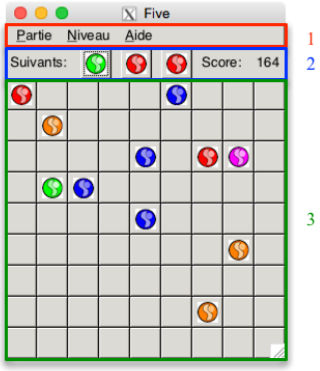


FIGURE 4 – Interface graphique générale du jeu.

FIGURE 5 – Boîte de dialogue"A Propos". indiquant la liste (triée) des 10 meilleurs scores.

FIGURE 6 – Boîte de dialogue

La Figure 4 présente l'IHM générale du jeu. Comme indiqué sur la figure, l'IHM est divisée en trois parties :

1. *Menu*. Le menu comprend trois sous-menus :
   (a) *Partie*. Ce menu permet de démarrer une nouvelle partie (avec le niveau de difficulté actuel), de consulter la liste des 10 meilleurs scores (cfr. Figure 5 pour un exemple d'une telle liste) ou bien de quitter le jeu ;
   (b) *Niveau*. Ce menu permet de sélectionner un des trois niveaux de difficulté du jeu (cfr. Sec. 2.3) ;
   (c) *Aide*. Ce menu permet d'obtenir via un item unique, *A propos*, ouvrant une boîte de dialogue avec des informations relatives sur le(s) créateur(s) du jeu. Un exemple de boîte de dialogue est donné à la Figure 6.
2. *Plateau d'information*. Ce plateau indique les prochains symboles [2] qui seront placés sur le plateau

2. Des images représentants les différents symboles sont disponible sur eCampus.

de jeu [3] et le score actuel du joueur.

3. *Plateau de Jeu.* Ce plateau contient les cases contenant les symboles à aligner lors du tour suivant.

6 Éléments Additionnels C

Cette section vous présente différents éléments additionnels du C et de GTK+2 qui peuvent vous être utiles pour l'implémentation du projet.

6.1 Image et Bouton

Il est possible, en GTK+2, d'afficher une image dans un bouton. Supposons que nous disposions d'une image, `img1.jpg`, et qu'on veut l'afficher dans un bouton `pBouton`. Ceci nécessite trois étapes :

1. Charger l'image depuis le disque dur. Éventuellement, on peut redimensionner l'image ;
2. Créer le bouton ;
3. Placer l'image dans le bouton.

Le bout de code ci-dessous illustre ces trois étapes.

```
GtkButton *charge_image_bouton(){
  //1a. Charger l'image
  GdkPixBuf *pb_temp = gdk_pixbuf_new_from_file("img1.jpg", NULL);
  if(pb_temp == NULL){
    printf("Erreur de chargement de l'image img1.jpg!\n");
    return NULL;
  }
  //1b. Redimensionner l'image en 100*100 pixels
  GdkPixBuf *pb = gdk_pixbuf_scale_simple(pb_temp, 100, 100, GDK_INTERP_NEAREST);
  if(pb == NULL){
   printf("Erreur lors de la redimension de l'image!\n");
   return NULL;
  }

  //2. Créer le bouton
  GtkWidget *pBouton = gtk_button_new();
  if(pBouton == NULL){
   printf("Erreur lors de la création du bouton\n");
   return NULL;
  }

  //3. Placer l'image
  GtkWidget *image = gtk_image_new_from_pixbuf(pb);
  if(image == NULL){
   printf("Erreur lors de la création de l'image\n");
   return NULL;
  }
  gtk_button_set_image(GTK_BUTTON(pBouton), image);

  return pBouton;
}//fin charge_image_bouton()
```

Si on désire remplacer l'image existante d'un bouton, il suffit d'appeler, sur ce bouton, la procédure `gtk_button_set_image` avec la nouvelle image.

Notez qu'à chaque fois que l'on remplace l'image du bouton, on crée un nouvel objet de type `GtkImage` [4] en appelant `gtk_image_new_from_pixbuf`. On pourrait supposer qu'il serait préférable de libérer l'image avant de créer la nouvelle. En réalité, lorsqu'on appelle `gtk_button_set_image`, GTK+2 libère la mémoire occupée par la `GtkImage` précédemment affichée.

3. 3 ou 7 symboles en fonction de la difficulté (voir Table 2).
4. A l'instar (par exemple) de `GtkWindow`, un objet de type `GtkImage` est aussi de type `GtkWidget`.

6.1.1 Dimensionnement des Boutons

Une fois un bouton créé (et, plus généralement, n'importe quel élément de type `GtkWidget *`), on peut toujours modifier sa taille par défaut. Il suffit d'utiliser la procédure suivante :

```
void gtk_widget_set_size_request(GtkWidget *wgt, gint largeur, gint hauteur);
```

qui redimensionne le widget `wgt` en un widget de `largeur` × `hauteur`. L'unité de mesure est ici le pixel.

6.2 Contrôler l'Aléatoire

En C, on peut produire une suite pseudo-aléatoire d'entiers en utilisant la fonction `rand()` de la bibliothèque standard (`stdlib.h`). A chaque fois qu'un programme appelle cette fonction, elle retourne l'entier suivant dans la suite. Les entiers de la suite sont compris entre `0` et `RAND_MAX` (une constante prédéfinie).

Par exemple, le programme suivant :

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

int main(){
  for(int i = 0; i < 10; i++){
    int a = rand();
    printf("%d\n", a);
  }//fin for - i

  return EXIT_SUCCESS;
}//fin programme
```

produit la sortie suivante :

```
16807
282475249
1622650073
984943658
1144108930
470211272
101027544
1457850878
1458777923
2007237709
```

Si on cherche à tirer aléatoirement un nombre entre 0 et N (**non** compris), il suffit d'utiliser le reste de la division par N :

```
int x = rand() % N;
```

Si on veut tirer un entier entre A et B compris, on peut faire :

```
int x = rand() % (B - A + 1) + A;
```

Pour produire un flottant entre A et B, on peut faire :

```
double x = (double) rand() / RAND_MAX * (B - A) + A ;
```

La suite produite par `rand()` n'est pas vraiment aléatoire. En particulier, si on lance le programme précédent plusieurs fois, on obtiendra exactement la même suite.

Pour rendre le résultat apparemment plus aléatoire, on peut initialiser la suite en appelant la fonction `srand()` (définie aussi dans `stdlib.h`), à laquelle on passe en paramètre une valeur d'initialisation. La suite produite par `rand()` après cet appel dépendra de la valeur que l'on passe à `srand()` lors de l'initialisation. Une même valeur d'initialisation produira la même suite.

Par exemple, le programme suivant :

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

int main(){
  srand(2);
```

```
6
7     for(int i = 0; i < 5; i++){
8       int a = rand();
9       printf("%d\n", a);
10    }//fin for - i
11
12    printf("\n");
13    srand(2);
14
15    for(int i = 0; i < 5; i++){
16      int a = rand();
17      printf("%d\n", a);
18    }//end for - i
19
20    return EXIT_SUCCESS;
21  }//fin programme
```

produit la sortie suivante :

```
33614
564950498
1097816499
1969887316
140734213

33614
564950498
1097816499
1969887316
140734213
```

Pour que la suite soit vraiment imprévisible, on peut passer à `srand()` une valeur dépendant de l'heure. Par exemple en utilisant la fonction `time()` de la librairie standard :

```
1 srand(time(NULL));
```

7 SCM

Dans le cadre de ce projet, il vous est aussi demandé d'utiliser un SCM (Source Code Manager). L'utilisation du SCM au sein de votre binôme interviendra dans la note globale de votre projet. Vous devrez, en outre, décrire la façon dont vous avez utilisé le SCM dans votre rapport (cfr. Sec. 8).

Le type de SCM à utiliser vous est imposé. Il s'agit de l'instance GitLab de l'Université. Elle est disponible à l'adresse suivante : https://gitlab.uliege.be/. Vous pouvez vous y connecter avec votre compte étudiant.

L'utilisation du GitLab nécessite une inscription préalable avec votre compte étudiant. Les deux membres du binôme doivent s'inscrire. Une fois que c'est fait, un des deux membres du binôme a la possibilité de créer un projet avec le niveau de visibilité **privé**. Le nom du projet, dans le GitLab, doit **obligatoirement** suivre la nomenclature suivante : `INFO0030_GroupeXX`, où `XX` désigne votre numéro de groupe [5] (il s'agit donc bien d'un nombre à deux chiffres). Le créateur du projet est alors libre d'ajouter l'autre membre du binôme au projet nouvellement créé. Vous devez **obligatoirement** ajouter les assistants @Emilien.Wansart1 et @Maxime.Goffart1 comme membres du projet avec le rôle "**maintainer**". Une fois cette étape passée, les deux membres du binôme (et uniquement eux) ainsi que les assistants peuvent accéder au répertoire de code.

L'URL associée à votre projet est disponible dans la partie supérieure droite de l'écran (une fois, bien entendu, que le projet a été créé). L'étape suivante consiste donc à cloner le projet en local sur votre machine en utilisant la commande :

```
1 $>git clone <URL>
```

où `<URL>` désigne l'URL de votre projet.

Pour rappel, voici quelques commandes qui pourraient vous être utiles [6] :

5. Veuillez utiliser le numéro de groupe qui vous a été attribué sur eCampus.
6. Nous vous renvoyons vers la formation Git et la documentation officielle pour plus de détails.

— Ajouter le fichier `<fichier>` à votre projet :

```
$>git add <fichier>
```

— Enregistrer les modifications et les envoyer sur le serveur GitLab :

```
$>git commit -m "Log Message"
$>git push
```

— Télécharger les modifications se trouvant sur le serveur GitLab :

```
$>git pull
```

8 Rapport

En plus du code source de votre application, nous vous demandons de joindre un rapport détaillé et clair expliquant :

— L'architecture générale de votre code. Quels sont les grands concepts de votre code et comment ils interagissent entre eux.

— Les structures de données. Vous serez amenés, dans ce projet, à développer des structures de données. Décrivez-les dans le rapport et pensez à discuter la pertinence et/ou le coût en mémoire de ces structures.

— Les algorithmes particuliers. Vous pourriez être amenés, dans ce projet, à développer des algorithmes poussés, pour le déroulement du jeu. Décrivez l'idée de ces algorithmes dans votre rapport et comment vous les avez implémentés (structures de données particulières, fonctionnement général, ...). [7]

— L'interface graphique du jeu. Fournissez des captures d'écran et commentez-les. Expliquez comment vous avez organisé votre jeu (table, box, ...).

— La gestion du code. Expliquez comment vous avez géré le SCM et ce que cela vous a apporté tout au long du projet. Veillez à ajouter le lien vers votre projet sur l'instance GitLab de l'Université (`https://gitlab.uliege.be/`).

— La coopération du sein du groupe. Expliquez comment votre binôme a fonctionné et comment vous avez géré votre coopération.

— Les améliorations possibles. Décrivez les améliorations possibles à votre application (par exemple, si vous aviez disposé d'un mois supplémentaire).

— Les éléments que vous avez appris. Décrivez ce que ce projet vous a apporté et ce que vous en avez appris.

Votre rapport doit être rédigé en **français** avec l'outil LaTeX. [8] Le document LaTeX doit suivre le template donné sur la page web du cours (eCampus). Vous penserez à prendre en compte les éléments abordés lors de la formation relative à la communication écrite.

Vous veillerez à joindre, dans votre archive (cfr. Sec. 9), les **sources** de votre rapport ainsi qu'une version PDF.

9 Énoncé du Projet

Contrairement aux autres projets, celui-ci doit être réalisé en binôme. A chaque groupe a été associé un identifiant (unique) sous la forme d'un nombre naturel à deux chiffres. [9]

Il vous est demandé d'écrire un programme implémentant le jeu Five or More tel que décrit dans ce document.

Votre projet devra :

7. Il n'est pas nécessaire d'indiquer les Invariants de Boucle, les spécifications formelles et, plus généralement, l'approche constructive dans votre rapport.
8. Nous vous renvoyons à votre formation LaTeXreçue en début de quadrimestre.
9. Voir le forum sur eCampus.

— Être soumis dans une archive `tar.gz`, appelée `five_XX.tar.gz`, via la Plateforme de Soumission, où `XX` désigne votre identifiant de groupe (n'oubliez pas de vous enregistrer en tant que groupe !). La décompression de votre archive devra fournir tous les fichiers nécessaires **dans le répertoire courant où se situe l'archive**. Vous penserez à joindre tous les codes sources nécessaires ainsi que votre rapport. Votre archive doit être chargée sur la Plateforme de Soumission pour le **Lundi 05/05/2025, 08h00** au plus tard.
— Définir les fonctionnalités nécessaires à la mise en place de l'IHM du Five or More. Des fichiers `PNG` vous sont fournis en supplément de façon à implémenter le jeu.
— Être modulaire, i.e., nous nous attendons à trouver un (ou plusieurs) header(s) et un (ou plusieurs) module(s). De manière générale, votre code devra impérativement suivre le pattern **modèle-vue-contrôleur** (MVC) tel que vu au cours. [10]
— S'assurer que les structures de données proposées sont implémentées comme des types opaques (quand cela s'avère pertinent).
— Être parfaitement documenté. Vous veillerez à spécifier correctement chaque fonction/procédure/énumération/structure de données que vous définirez. Votre documentation suivra les principes de l'outil `doxygen`.
— Appliquer les principes de la programmation défensive (vérification des préconditions, vérification des mallocs, ...). Pensez à libérer la mémoire allouée en cours de programmation afin d'éviter les fuites de mémoire.
— Utiliser un SCM (cfr. Sec. 7).
— Contenir un rapport rédigé en **français** avec l'outil LaTeX. Vous veillerez à fournir les sources de votre rapport ainsi que la version compilée en PDF (voir Sec. 8 pour le format attendu).
— Contenir un `Makefile` permettant de :
  * Compiler votre programme. La commande

```
$>make five
```

    doit pouvoir compiler votre programme et générer un binaire exécutable appelé `five`.
  * Générer la documentation, au format HTML, en suivant les principes de `doxygen`. La commande

```
$>make doc
```

    doit permettre la génération de la documentation dans un sous-répertoire `doc/`.
  * Générer le PDF de votre rapport. La commande

```
$>make rapport
```

    doit permettre cela.
— Pouvoir exécuter le binaire `five` de la façon suivante :

```
$>./five -n <difficulty> -f <file> [-h]
```

où :
  * `-n` : niveau de difficulté <difficulty> peut prendre les valeurs 0 (facile), 1 (normal) ou 2 (difficile).
  * `-f` : chemin vers le fichier <file> maintenant les scores.
  * `-h` : aide pour la ligne de commande.

Si nous appliquons la commande suivante à votre archive :

```
$>tar xvfz five_XX.tar.gz
```

l'architecture suivante de répertoires doit apparaître :

```
GroupeXX
  |
  ---source/
  ---doc/
  ---rapport/
```

10. cfr. Partie 3, Chapitre 3, du cours théorique

où le sous-répertoire `source/` contient les fichiers sources de votre programme, le sous-répertoire `doc/` contient la documentation au format HTML déjà générée par `doxygen` (pensez à bien mettre une règle Makefile pour que nous puissions la regénérer dans ce sous-répertoire) et, enfin, le sous-répertoire `rapport/` contient les sources de votre rapport et le fichier PDF correspondant. Quant au Makefile, il se trouve à la racine du dossier `GroupeXX`.

Attention, le projet ne se termine pas à la soumission de votre archive. En plus de tout cela, vous devrez réaliser une **démonstration de votre projet** devant l'équipe pédagogique. Cela consistera en une exécution live (**maximum 5 minutes par groupe**) de votre application. Durant cette exécution, vous expliquerez, oralement, le contenu de votre rapport et vous démontrerez que votre jeu fonctionne parfaitement. Après les 5 minutes de démonstration, l'équipe pédagogique disposera (éventuellement) de 5 minutes afin de vous poser des questions pour clarifier certains points. L'ordre de passage pour la démonstration sera donné sur la page web du cours.

10 Cotation

Étant donné le caractère inhabituel et plus complet de ce projet par rapport aux autres, la grille de cotation que nous allons appliquer sera légèrement différente :

— 50% de la note du projet sera affectée suivant la nomenclature classique adoptée dans le cours.
— 10% de la note du projet portera sur la façon dont vous avez utilisé votre SCM.
— 10% de la note du projet portera sur l'application du pattern MVC.
— 15% de la note du projet portera sur le rapport écrit (qualité globale du document [11], orthographe, qualité et précision des explications, esprit de synthèse, ...).
— 15% de la note du projet portera sur la démonstration live de votre jeu (qualité du jeu, clarté de la présentation, gestion du temps, scénarisation de la démonstration, réponses aux questions, ...).

Toute question relative au projet peut être posée sur le forum de la page web du cours (eCampus).

Tout projet ne compilant pas se verra attribuer la note de 0/20.

Il est impératif de respecter **scrupuleusement** les consignes sous peine de se voir attribuer une note de 0/20 pour non respect de l'énoncé.

Tout groupe ne soumettant pas à minima une archive vide se verra attribuer une note d'absence.

11. Nous vous référons à la formation que vous avez reçu en début de quadrimestre sur la façon de rédiger un rapport.