# Projet Structure de Données Avancées

## Objectif

Les dungeon cawlers sont des jeux (de société ou vidéo) qui mettent l'accent sur l'exploration de donjons. Dans cette sorte de jeux un donjon se concrétise par une carte sur laquelle figure des informations importantes comme des *portes*, des *monstres* et des *trésors*. On souhaite développer quelques bibliothèques C rudimentaires *simplifiant* la gestion de donjons. Pour notre application, les donjons seront représentés par des grilles dont chaque case peut contenir au plus un élément. Voici une liste non exhaustive des éléments qui peuvent apparaitre dans un donjon :

- Un mur ;
- Un monstre ;
- Une dépouille (quand un monstre est mort) ;
- Un coffre ;
- Un piège ;
- Une porte ouverte ;
- Une porte fermée ;
- Un champion (Un monstre plus important dans l'intrigue ou simplement plus fort) ;
- Un autel ;
- Une entrée ;
- Une sortie (Dans la vie quotidienne est possible de sortir par une entrée mais pour un donjon les sorties mènent généralement à de nouveaux niveaux).

Par souci de simplicité, vous considérerez uniquement des déplacements[1] guidés par les points cardinaux : NORD, SUD, EST et OUEST.

Un concepteur de donjon procédera par étapes :

1. Il pourra créer des salles ou des couloirs ;
    a. Lorsqu'il créera une salle, il commencera par la dimensionner puis placera des éléments à l'intérieur de cette salle.
    b. Lorsqu'il créera un couloir, il commencera par indiquer le nombre de déplacements nécessaires pour le traverser, puis il indiquera la séquence de placements.
2. Il pourra créer des donjons en assemblant des salles et des couloirs. Le plan d'une salle pourra être *recyclé* dans plusieurs donjons ou même apparaître plusieurs fois dans un même donjon.
    a. Je vous suggère de repérer la position d'une salle les coordonnées de son coin supérieur gauche (par exemple).
    b. Vous prendrez garde à ce que l'ajout d'une salle rentre effectivement dans la carte d'un donjon. Si le placement est impossible le designer de donjon devra être informé de cette impossibilité et le placement d'annuler.
    c. Si une salle est placée sur des éléments déjà présent sur la carte ces derniers seront perdus.
3. Votre application permettra notamment de
    a. Visualiser un donjon en mode texte (voir figure 1).
    b. Créer un donjon
        i. Le dimensionner ;
        ii. Placer des salles ;

[1] Pour le moment, il n'est pas encore question de faire bouger des personnages dans le donjon.

iii. Placer des couloirs ;
iv. Modifier directement un élément du donjon ;
v. Sauvegarder le donjon dans un fichier.

c. Créer des salles :
   i. Dimensionner la salle ;
   ii. Placer des éléments dans la salle ;
   iii. Sauver une salle dans un fichier.

d. Créer des couloirs :
   i. Dimensionner le couloir ;
   ii. Renseigner les déplacements le long du couloir ;
   iii. Sauver un couloir dans un fichier.

sujet_projet > exemple_donjon.txt

```
50
70
###############E#####################################
#            # #                                     #
#      ########O#########                            #
#      #                #        ######             #
#      #    #     #     #        #  T#               #
#      #                ##########   ########       #
#   ####       M        O                    #       #
#   #  O     #     #     ##########   ###### #       #
#   # ##           M     #        ### #    # #       #
#   # ##              C#           # #    # #       #
#   # #########F#########           # #    # #       #
#   # # ###### #      ############## #    # #       #
#   # # #      #      #              #    # #       #
#   # # # ######      # ##############    # #       #
#   # # # #     ########O#########        # #       #
#   # # # #     #                #        # #       #
#   # # # #     #    #     #     #    ######F#      #
#   # # # #     #                ######     #       #
#   # # # #######       M       O    F  A  #       #
#   # # #       O     #     #    ######     #       #
#   # # #########           M    #    #######       #
#   # #          #              C#                   #
#   # #          ########F#########                  #
#   # #                #    #                        #
#   # #            ######## #                        #
#   # #            #        #                        #
#   # #            # ########                        #
#   # #            # #                               #
#   # #      #######O##########   ##########         #
#   # #      #                #   #C       C#        #
#   ## #     #    #     #     #   #   M    #         #
#   #  #     #                #####   MM   #         #
#   # ########       M        O   F   MM   #         #
#   #         O     #     #    #####   M    #         #
#   ##########           M    #   #        #         #
#            #              C#   #PW      A#         #
#            ########F#########   ##########         #
#                   # #############                  #
#                   #            #                  #
```

**Figure 1** - Exemple de fichier contenant un donjon. Dans ce fichier, `'#'` représente un mur, `'C'` représente un coffre, `'M'` un monstre, `'W'` une dépouille, `'O'` une porte ouverte, `'F'` une porte fermée, `'A'` un autel, `'P'` un piège, `'B'` le champion, `'E'` une entrée et `'S'` une sortie. Les dimensions du donjon sont fournies au début du fichier. On remarque dans ce donjon la présence de trois fois la même salle. On voit aussi la dépouille d'un monstre à côté d'un piège.

## Compléments

Le projet doit être développé en C standard sans ajout de bibliothèques supplémentaires (notamment graphiques). Cependant, vous êtes libre de la forme que peut prendre la solution et les objectifs décrits présentent le projet sous une forme nominale (c'est-à-dire une version minimale de l'application). Vous serez aussi évalué :

- Sur les ajouts que vous apportez à l'application et à votre bibliothèque ;
- Sur la propreté des sources : **modularité**, commentaires, choix des noms des fonctions et des variables ;
- Le rapport ;
- En annexe dans le rapport ou dans un fichier README.txt à part, vous présenterez l'installation et le fonctionnement du programme.

## Mise en garde

La formulation de ce document est sujet à interprétations. Il **vous** incombe de vous réaliser ces interprétations de manière cohérente avec les divers morceaux de votre programme. Ces interprétations pourront être négocier avec le *professeur* mais dans tout état de causes les choix retenus seront présentés et discutés dans le rapport. Dans le projet de cette année, le professeur ne fournis pas de fichiers .h, vous êtes libres de construire les types que vous jugez utiles : Plusieurs solutions restent possibles. Certaines sont plus simple à implanter d'autres sont plus efficaces beaucoup d'entre-elles ne sont ni simples ni efficaces. Donc commencer tôt pour vous laissez le temps de vous tromper et de recommencer avant la fin du projet.

Les différents types seront développés dans des **bibliothèques séparées**. Vous fournirez avec votre projet :

- Les sources : les fichiers .c et .h nécessaires à la compilation du programme.
- Un fichier `makefile` permettant de compiler les bibliothèques et un programme principal permettant de tester ensemble de vos routines et tous vos types.
- Un rapport qui présente notamment :
    - Le problème que vous vous apprêtez à résoudre ;
    - Les choix que vous avez fait pour résoudre le problème ;
    - La solution que vous apportez au problème que vous avez circonscrit.
- Un README.txt (si les informations ne figurent pas dans le rapport)

Vous trouverez un kit de survie sur l'écriture de rapport.

L'ensemble de vos fichiers seront déposés sur icampus dans une **archive zip** avant la **date spécifiée** par votre enseignant sur la plateforme.

Vous êtes invités à proposer des ajouts et des améliorations sur votre projet. Cependant, prenez le temps de valider les fonctionnalités de base du projet avant de proposer ces upgrades. Les éventuels ajouts ne compteront pas si la version de base du projet n'est pas opérationnelle.

## Un dernier mot

**Assurez-vous que votre produit compile avant de rendre le projet et que toutes les fonctions développées sont opérationnelles !**

Courage aventurier, vous pénétrez dans l'*antre de la Bèèèète*.