INFO-F101 – Projet du cours de Programmation
Exploration de Donjon

Année académique 2024-2025

Introduction

Dans ce projet, vous allez développer un jeu d'exploration de donjon. Le joueur incarne un aventurier dont l'objectif est de collecter des trésors tout en affrontant des monstres. Ce jeu est proposé avec trois niveaux de difficulté.

Vue d'ensemble

Avant de débuter une partie, le joueur a le choix entre trois niveaux de difficulté. Chaque niveau offre une expérience de jeu différente, mais le but du jeu reste le même : trouver les trésors. Un donjon carré de taille N×N, avec N toujours supérieur ou égal à 4, est chargé avec les spécificités de la difficulté choisie. Votre personnage entre toujours dans le donjon par la case située dans le coin supérieur gauche, comme indiqué dans la figure figure 1, où votre personnage est représenté par la lettre `P`.
Le donjon est peuplé de monstres qui sont mobiles ou non en fonction du niveau de difficulté. Votre personnage débute la partie avec un certain nombre de points de vie qui dépend du niveau de difficulté.
Avec de simples déplacements (haut,gauche,bas,droite), votre personnage doit tenter de trouver tous les trésors pour terminer la partie et gagner.
Une partie se termine lorsque les points de vie de votre personnage tombent à 0 (il meurt et c'est une partie perdue) ou lorsqu'il a trouvé tous les trésors (partie gagnée).

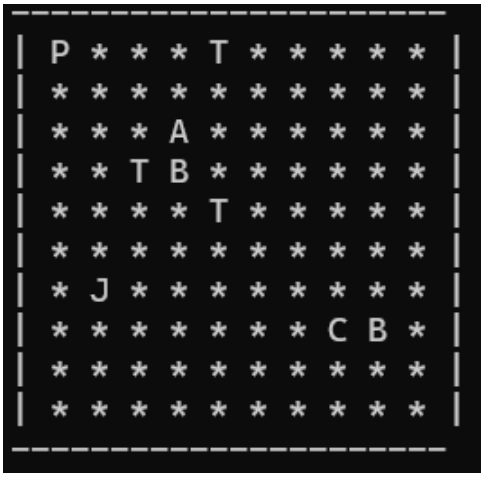

Figure 1 – Un donjon en difficulté facile avec les trésors visibles

Chargement du donjon

Quel que soit le niveau de difficulté, les spécifications de la carte se trouvent dans le fichier `carte.txt`, situé dans le même répertoire que votre code source. Les symboles, valeurs et coordonnées peuvent différer d'un fichier à l'autre, mais le format reste toujours conforme à ce qui est indiqué sur la figure figure 2 et respecte les contraintes de l'énoncé.

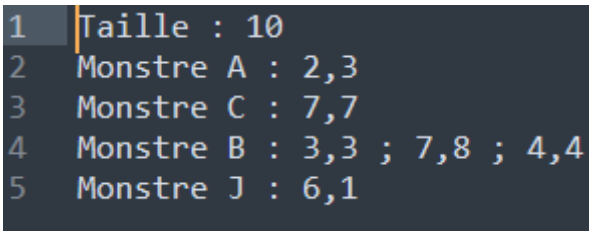

Figure 2 – Exemple de fichier `carte.txt`

Ce fichier comporte plusieurs informations :

- La taille du donjon à générer. Le donjon est toujours un carré NxN avec N >= 4.
- Les types de monstres qui se trouvent dans le donjon ainsi que leurs positions.

Les positions des différents monstres de même type sont séparées par un point-virgule comme illustré dans la figure 2 pour les Monstres B.

Type de monstre

Un type de monstre est défini par ses dégâts et son symbole. Les monstres sont symbolisés par une lettre Majuscule allant de A jusque J. Le jeu est donc limité à 10 types de monstres différents.
Les dégâts d'un monstre sont définis par la formule suivante :

```
3 * position_alphabet_symbole

Monstre A = 3 * 1 = 3 dégâts
Monstre J = 3 * 10 = 30 dégâts
```

Niveau 0 (Facile)

- **Points de vie** : 100
- **Monstres** : Les monstres ne se déplacent pas et disparaissent une fois qu'ils ont combattu avec le personnage.
- **Trésors** :
  - 3 trésors répartis aléatoirement au lancement du jeu sur les cases non occupées
  - Les trésors sont visibles

Niveau 1 (Moyen)

- **Points de vie** : 20
- **Monstres** :
  - À chaque tour, après le déplacement du joueur, tous les monstres ayant le plus haut score d'attaque à l'initialisation du jeu se déplacent aléatoirement vers une des quatre cases adjacentes : haut, gauche, bas ou droite. Cette case doit être inoccupée ou occupée par le personnage. Si un monstre se déplace vers le personnage, un combat s'engage, le monstre inflige des dégâts au joueur, puis

disparaît du donjon. Dans la figure 2, seulement le monstre de type J se déplace, une fois ces monstres là vaincus, il n'y aura plus de déplacement de monstre durant cette partie.
  - Les autres monstres sont immobiles.
- **Trésors** :
  - 6 trésors répartis aléatoirement sur des cases inoccupées
  - Les trésors sont cachés et donc ne sont plus visibles lors de l'affichage de la carte

Niveau 2 (Difficile|WTF !)

- **Points de vie** : 10
- **Monstres** :
  - Après le déplacement du joueur, tous les monstres se déplacent vers une case adjacente ou diagonale. Toujours sur une case inoccupée ou occupée par le personnage.
  - Tous les 5 tours, un nouveau monstre avec des dégâts aléatoires parmi les 10 types de monstres est ajouté dans le donjon, sur une case non occupée. Lors de l'ajout de ce monstre, un message s'affiche pour préciser son symbole, les dégâts calculés à partir de ce symbole, ainsi que sa position comme lors du lancement du jeu illustré dans la figure 4.
- **Trésors** : 10 trésors cachés répartis aléatoirement sur des cases inoccupées.
- **Soins** : Tous les 3 tours, une trousse de secours apparaît sur une case inoccupée du donjon. Ramasser cette trousse de secours ajoute 5 points de vie au personnage. Un message est affiché lors de l'apparition d'une trousse de secours. La trousse de secours est visible pour le joueur et est symbolisé par le symbole : +.
- **Pièges** : Au début de la partie, trois pièges sont placés aléatoirement dans le donjon sur des cases non occupées. Ces pièges réduisent les points de vie du personnage à 0 lorsqu'il marche dessus et sont invisibles pour le joueur. Après un déplacement du personnage, si un piège se trouve sur une case adjacente ou diagonale à la nouvelle position du personnage, un message d'avertissement est affiché.

Tour de jeu

À chaque tour, le joueur :

1. Se déplace vers une nouvelle case avec les 4 possibilités : haut,gauche,bas,droite. La première lettre du déplacement sera demandé à l'utilisateur (h,g,b,d). S'il donne une commande différente ou impossible à réaliser (comme g à la position initiale), le jeu lui demande de refaire un choix parmi les 4 déplacements : (h,g,b,d).

```
Déplacez-vous (h: haut, b: bas, g: gauche, d:droite) : PGKFH
Seulement 4 déplacements possibles (h: haut, b: bas, g: gauche, d: droite :
```

Figure 3 – Déplacement

2. Peut rencontrer un trésor, un monstre ou un piège.
3. S'il rencontre un trésor, il le récupère.
4. S'il rencontre un monstre, il le combat et s'inflige des dégâts en fonction du monstre combattu. Le monstre disparaît alors du donjon.
5. S'il rencontre un piège, il est notifié et perd la partie.

6. Doit voir les points de vie du personnage ainsi que le nombre de trésors restants à la fin de chaque tour.

Les tours de jeu peuvent varier selon la difficulté. L'énumération ci-dessous donne l'idée générale du déroulement d'un tour. Les effets tel que l'apparition de pack de soin ou monstre se déroule au début du tour.

Initialisation du jeu

Quand on lance le programme, la première chose demandée au joueur est la difficulté choisie. La difficulté est un chiffre allant de 0 à 2 respectivement pour le mode Facile, Moyen, Difficile|WTF ! comme sur la figure 4.

```
### Nouvelle partie initiée ! ###
Bienvenue dans l'exploration de donjon !
Choisissez la difficulté (0: Facile, 1: Moyen, 2: Difficile) : 0
1 monstre A avec 3 dégâts est ajouté à l'emplacement (2,3)
1 monstre C avec 9 dégâts est ajouté à l'emplacement (7,7)
3 monstres B avec 6 dégâts sont ajoutés aux emplacements (3,3),(7,8),(4,4)
1 monstre J avec 30 dégâts est ajouté à l'emplacement (6,1)
```

Figure 4 – Lancement du jeu

Affichage

L'affichage du jeu se fait de manière textuelle. Les contours de la carte sont encadrés comme illustré dans la figure 5. Chaque case du donjon est représentée par un caractère en fonction de la case en question

- Une case non occupée : *
- Votre personnage : P
- Trésors : T
- Packs de soin : +
- Les monstres : A-B-C...-J Ordre alphabétique pour les symboles des 10 types de monstres.
- La case (0,0) est le coin supérieur gauche, (0,1) est à celle à sa droite, (1,0) est celle en dessous.
- Les positions sont données par le tuple (ligne,colonne)

Voici le donjon chargé pour le fichier `carte.txt` de l'exemple ci-dessus :

```
### Nouvelle partie initiée ! ###
Bienvenue dans l'exploration de donjon !
Choisissez la difficulté (0: Facile, 1: Moyen, 2: Difficile) : 1
3 monstres A avec 3 dégâts sont ajoutés aux emplacements (2,3),(2,2),(1,1)
1 monstre E avec 15 dégâts est ajouté à l'emplacement (4,2)
--------------
| P * * * * * |
| * A * * * * |
| * * A A * * |
| * * * * * * |
| * * E * * * |
| * * * * * * |
--------------
Vie(s) : 20, Trésor(s) restant(s) : 6
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) :
```

Figure 5 – Donjon d'un autre fichier `carte.txt`, les trésors ne sont pas visibles dans ce niveau de difficulté.

Constantes d'affichage : Messages.txt

Un fichier MESSAGES.txt vous est fourni contenant des constantes à utiliser pour l'affichage. Vous pouvez changer les noms de variables utilisés dans les f-strings, voir figure 6. Pour l'affichage, vous devez suivre exactement les indications données dans la section implémentation.

```
#A utiliser dans votre code
MESSAGE_BIENVENUE = "### Nouvelle partie initiée ! ###\nBienvenue dans l'exploration de donjon !\n"

MESSAGE_INPUT = "Choisissez la difficulté (0: Facile, 1: Moyen, 2: Difficile) : "
MESSAGE_INPUT_ERREUR = "Seulement 4 déplacements possibles (h: haut, b: bas, g: gauche, d: droite : "

MESSAGE_PIEGE = "Attention ! Nous avons détecté {pieges_adjacents} piège(s) dans les cases adjacentes ou diagonales."

MESSAGE_AJOUT_UN_MONSTRE = f"1 monstre {symbole_monstre} avec {degats_monstre} est ajouté à l'emplacement {position_monstre}"

MESSAGE_AJOUT_DES_MONSTRES = f"{nombre_monstre} monstres avec {degats_monstre} sont ajoutés aux emplacements {position_monstre}"

MESSAGE_VICTOIRE = "Félicitations, vous avez collecté tous les trésors !"

MESSAGE_DEFAITE = "Vous êtes mort... Game over !"

MESSAGE_RENCONTRE = f"Vous rencontrez un monstre {monstre_combattu_symbole} ! Vous perdez {monstre_combattu_degats} points de vie."


#A utiliser dans afficher_grille

MESSAGE_VIE_TRESOR = f"Vie(s) : {vie_personnage}, Trésor(s) restant(s) : {tresors_restants}" #Vie & Trésors restants
```

Figure 6 – Contenu des constantes et f-string de `MESSAGES.txt`.

Explication des pièges du mode difficile|WTF !

```
Vie : 100, Trésors restants : 3
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) : d
----------------------
| * P * * T * * * * * |
| * * * * * * * * * * |
| * * * A * * * * * * |
| * * T B * * * * * * |
| * * * * T * * * * * |
| * * * * * * * * * * |
| * J * * * * * * * * |
| * * * * * * * C B * |
| * * * * * * * * * * |
| * * * * * * * * * * |
----------------------
Vie : 100, Trésors restants : 3
Attention ! Nous avons détecté 1 piège(s) dans les cases adjacentes ou diagonales.
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) :
```

Figure 7 – Le joueur arrive à la case (0,1) et est notifié de la présence d'un piège dans une case adjacente ou diagonale.

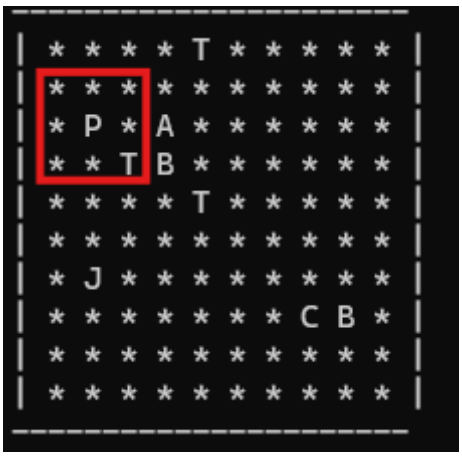

Figure 8 – Les cases adjacentes ou diagonales dans lequel il pourrait y avoir un piège. La case (3,2) étant occupée par un trésor, il ne peut y avoir de piège sur cette case.

Exemple d'exécution

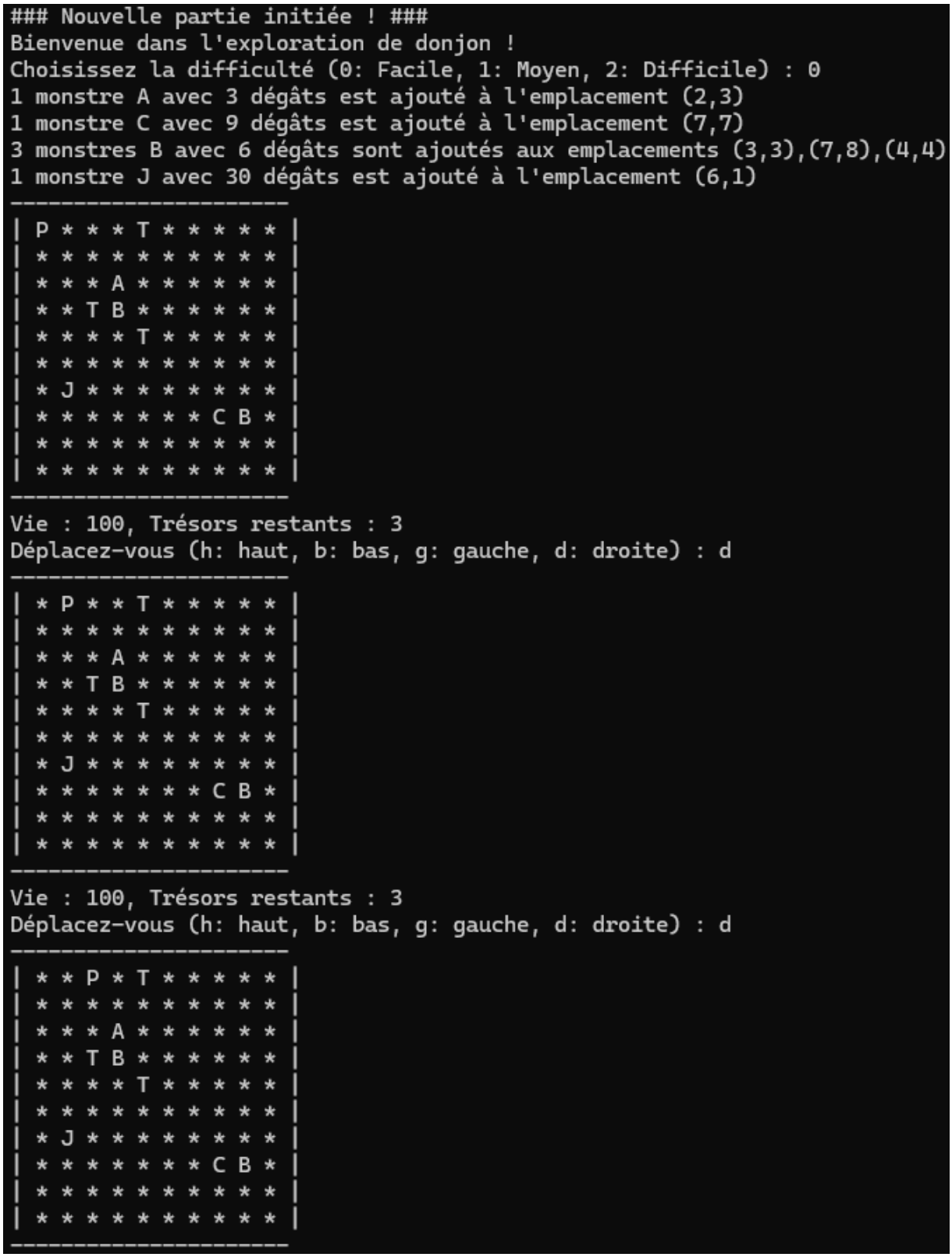

Figure 9 – Déroulement d'une partie en difficulté Facile 1/4

```
Vie : 100, Trésors restants : 3
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) : d
----------------------
| * * * P T * * * * * |
| * * * * * * * * * * |
| * * * A * * * * * * |
| * * T B * * * * * * |
| * * * * T * * * * * |
| * * * * * * * * * * |
| * J * * * * * * * * |
| * * * * * * * C B * |
| * * * * * * * * * * |
| * * * * * * * * * * |
----------------------
Vie : 100, Trésors restants : 3
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) : d
Vous avez trouvé un trésor !
----------------------
| * * * * P * * * * * |
| * * * * * * * * * * |
| * * * A * * * * * * |
| * * T B * * * * * * |
| * * * * T * * * * * |
| * * * * * * * * * * |
| * J * * * * * * * * |
| * * * * * * * C B * |
| * * * * * * * * * * |
| * * * * * * * * * * |
----------------------
Vie : 100, Trésors restants : 2
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) : b
----------------------
| * * * * * * * * * * |
| * * * * P * * * * * |
| * * * A * * * * * * |
| * * T B * * * * * * |
| * * * * T * * * * * |
| * * * * * * * * * * |
| * J * * * * * * * * |
| * * * * * * * C B * |
| * * * * * * * * * * |
| * * * * * * * * * * |
----------------------
Vie : 100, Trésors restants : 2
Déplacez-vous (h: haut, b: bas, g: gauche, d: droite) : b
----------------------
| * * * * * * * * * * |
| * * * * * * * * * * |
| * * * A P * * * * * |
| * * T B * * * * * * |
| * * * * T * * * * * |
| * * * * * * * * * * |
| * J * * * * * * * * |
| * * * * * * * C B * |
| * * * * * * * * * * |
| * * * * * * * * * * |
----------------------
```

Figure 10 – Déroulement d'une partie en difficulté Facile 2/4

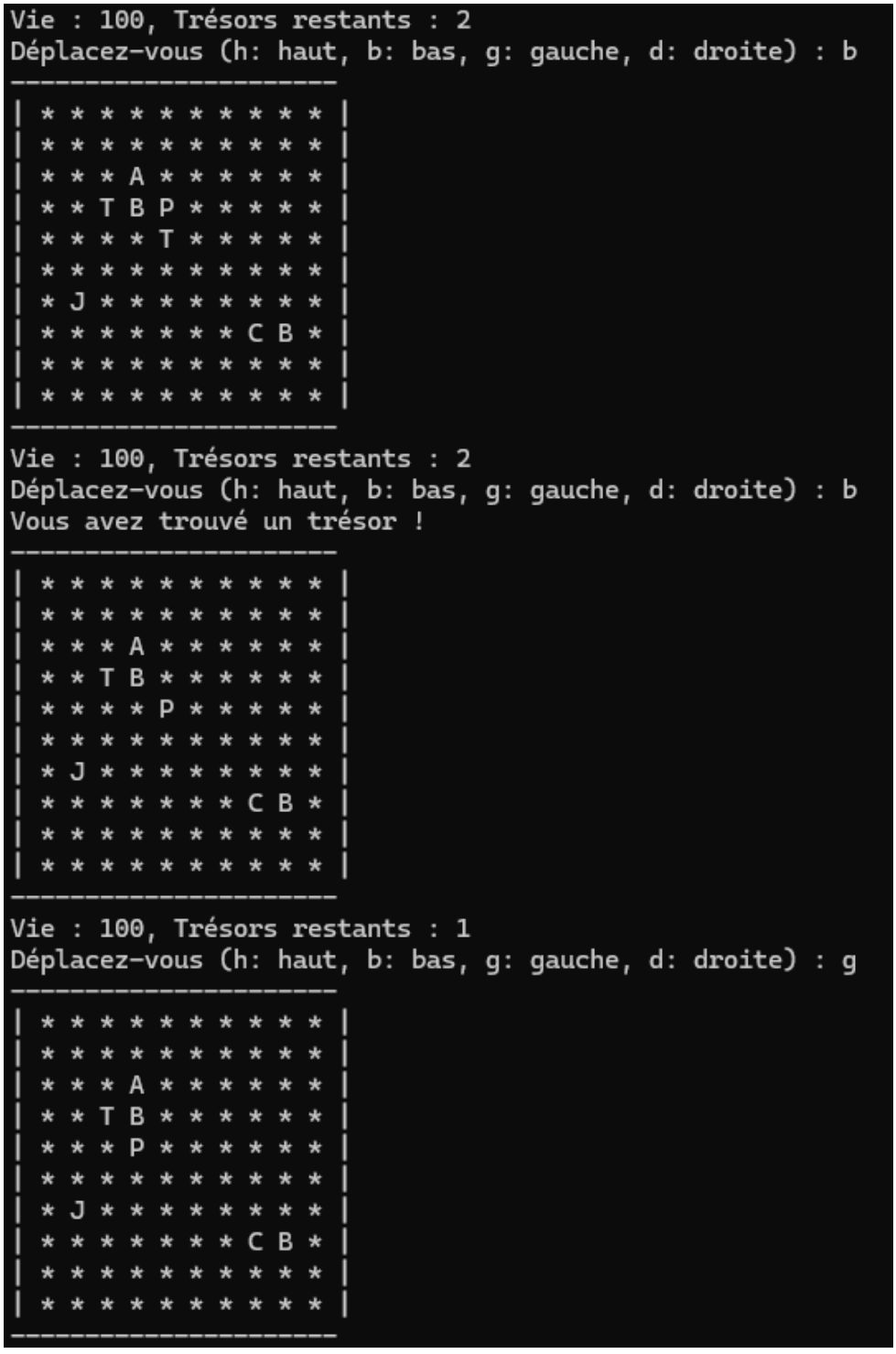

Figure 11 – Déroulement d'une partie en difficulté Facile 3/4

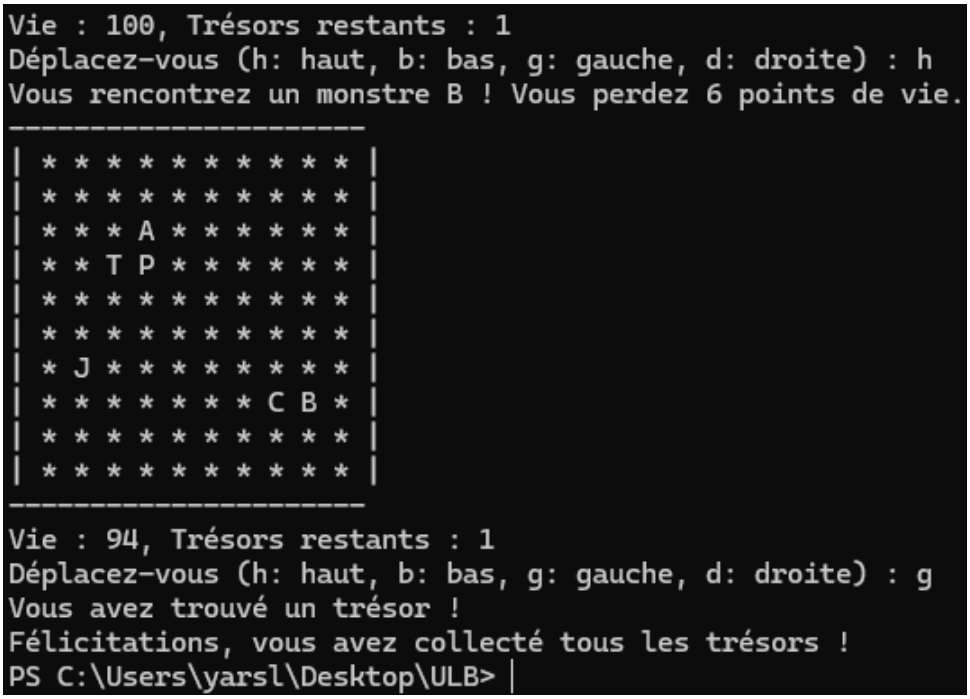

Figure 12 – Déroulement d'une partie en difficulté Facile 4/4

Implémentation

Il vous est demandé dans ce projet d'implémenter les fonctions avec les signatures suivantes :

— `lire_carte(fichier: str) -> dict` : Ouvre et lit le fichier dont le chemin est reçu en paramètre. Récupère, dans **un dictionnaire**, la taille du donjon et les positions des monstres groupées par leur symbole. Un exemple de résultat de la fonction est donnée dans la figure 13.

```
{"Taille" : 5, "A" : [(2,3)], "C" : [(7,7)], "B" : [(3,3), (7,8), (4,4)], "J" : [(6,1)]}
```

Figure 13 – Lire_carte sur le fichier de la figure 2

— `grille_string(grille) -> str` : Renvoie une représentation de la grille sous forme de string. Ce résultat est sensible à la casse [1] et doit être exactement similaire aux figures de ce document. Un extrait de code est fourni dans la figure 16 pour vous aider.

— `afficher_grille(grille, vie: int, tresors_restants: int)` -> None : Affiche la grille (en appelant la fonction afficher_grille) et affiche le résultat (points de vie et trésor(s) restant(s))

— `deplacer_personnage(direction, position_personnage, grille, vie) -> tuple` : Déplace le joueur sur la carte et renvoie un tuple (position_personnage, grille, vie)

1. Se dit de tout programme qui fait une distinction entre les lettres majuscules et les lettres minuscules.

```
ajout_cadre = 2
formule_cadre = 2
print("-" * (taille_donjon * formule_cadre + ajout_cadre))  # Bordure du haut

# Traitement de chaque ligne
# CODE A AJOUTER ICI

print("-" * (taille_donjon * formule_cadre + ajout_cadre))  # Bordure du bas
```

Figure 14 – Affichage de la grille sensible à la casse.

Ces fonctions et signatures doivent impérativement être utilisées dans votre code. Si le type n'est pas précisé, vous êtes libre d'utiliser la structure de donnée que vous souhaitez. Pensez à utiliser la structure de donnée adaptée à votre besoin.
La liste de fonctions donnée est non-exhaustive ; il vous est demandé d'implémenter davantage de fonctions tout en respectant les bonnes pratiques de programmation vues dans **le MOOC**, **le cours** et **les travaux pratiques**, notamment les noms de variables, les nombres magiques, l'utilisation de structures de données adaptées, la bonne utilisation des ressources et la découpe de votre programme en fonctions.

Module Random

Pour la génération de l'aléatoire, nous vous demander d'importer le module random et d'utiliser la fonction randint(x : int, y : int) -> int qui génère un nombre aléatoire entre [x ;y].

```
1  from random import randint
2
3  randint(0,5) # Retourne une valeur entre [0,5], bornes comprises
```

Figure 15 – Fonction randint

Nous vous demandons également d'importer et utiliser la fonction seed(x : int) -> None pour initialiser le générateur de nombres d'aléatoires.
Lors de la remise de votre projet, la seed utilisée doit être la seed(10).

```
1  from random import seed
2
3  seed(10) # Cette ligne doit être ajoutée une seule fois au début de votre code
```

Figure 16 – Fonction seed

Consignes pour la remise du projet

Les consignes de remise du projet sont fournies sur la page du cours. Il est essentiel de suivre ces instructions avec attention. Ce projet est à faire **SEUL**, vous avez le droit de discuter du projet mais il est interdit de partager ni votre code ni vos algorithmes. Le scanner de plagiat détectera le plagiat et les mesures nécessaires seront prises.
Vous devez remettre un seul fichier sous forme : **NOM_PRENOM.py** contenant votre code.
Pour toute question, vous pouvez contacter l'enseignant responsable : ARSLAN Yasin.
Mail : yasin.arslan@ulb.be

Date limite de remise : 18 novembre 2024 8h00