Algorithmique et langage C

Projet - Le labyrinthe éternel

Introduction et spécifications du projet

Dans un lointain pays existait un labyrinthe. Le Labyrinthe éternel était un lieu d'entrainement aux cent formes et mille dangers. Il est dit que celui qui en verra le bout deviendra immensément riche, mais personne n'a jamais réussi à en ressortir vivant.
Les trésors réservés au gagnant de l'épreuve sont encore là, et les portes restent ouvertes aux plus courageux - ou téméraires - qui voudront tenter leur chance !

Le but de ce projet est de mettre en place un jeu qui modélisera un personnage dans son exploration du labyrinthe. Le jeu est conçu pour être être un jeu de réflexion, demandant au joueur d'approcher les problèems dans le bon ordre afin de ressortir victorieux.
Cela impliquera des déplacements, une gestion d'inventaire, des combats. De plus plus, le sujet demande un système de gestion de niveaux par fichiers.

Ce projet est à développer avec groupe de 4 personnes. Il est conseillé de réfléchir à la répartition de la charge et d'utiliser des outils de mise en commun du travail (Git, communications de groupe, ..).
Le cadre de ce projet étant l'enseignement C, le seul langage autorisé est le C et les scripts de compilation makefile (pas de C++, notamment). Le programme devra pouvoir être compilé sur un ordinateur de démonstration fourni par le groupe lors de la soutenance, à l'aide de gcc ou d'un makefile approprié (cmake ne sera pas autorisé pour ce point spécifique, notamment).

En plus du code qui est à rendre, il sera demandé au groupe une rapide présentation du programme qui fera intervenir chaque membre (supports éventuels laissés au choix du groupe).

Il est recommandé de lire le sujet entièrement avant de commencer le développement afin d'anticiper d'éventuelles contraintes de code.

Description fonctionnelle

Partie 1 - Le monde et le personnage

Le monde dans lequel le personnage va évoluer est divisé en salles. Chaque salle correspond à un ensemble de cases en 30 lignes et 30 colonnes (pour un total de 900 cases par salle). Chaque case ne peut contenir qu'une seule chose à la fois :

- Un mur intraversable
- Une porte
- Un objet
- Une clé
- Une potion
- Un powerup
- Le personnage
- Un adversaire
- Un passage vers une autre salle

Comme chaque case ne peut comporter qu'une seule chose à la fois, il est possible de simplement afficher en console un tableau de 30x30 caractères en fonction du contenu de la case. Exemple (inspiré de nethack) :

```
#################^^################
#                                 #
# #############      ############ #
# #2C #   # A #      #  #A  #123# #
# #C  # B #   #        §#   # A # #
# #   #   #   #      #  #   #   # #
# #   #   ## ##      ##### ###o## #
# ##o### ##                        #
<                                 >
# ## ### ##     ###o###           #
# #   |   ##### #3   §# ###### ## #
# #   |   #!! # #  B  # #  #    # #
# #   |  A#!    # B1B #   A#    # #
# # A |   #   # #  B  # #A # C  # #
# ############# ####### ######### #
#                                 #
#################vv################
```

Dans cet espace, le personnage peut se déplacer dans les cases adjacentes (haut, bas, gauche, droite, pas en diagonale) si le terrain le permet. Les murs ne permettent pas le déplacement. Se déplacer vers un adversaire initie un round de combat (cf partie 2) tandis qu'un déplacement vers une porte ou un objet requiert une gestion plus spécifique (cf partie 3).
Se déplacer vers l'extéreur de la salle (les symboles < > v ^dans l'exemple) fait passer le personnage dans une nouvelle salle si elle existe.
Les différentes salles sont liées les unes aux autres dans une plus grande grille qui corresponds à l'ensemble du niveau. Cette grille est faite de ligne et colonnes qui peuvent contenir ou pas une salle.

Les passages vers une autre salle se trouvent sur les bords de la salle et se distinguent selon leur position : mur est, sud, ouest ou nord. Lors que le personnage arrive à un tel passage, il est amené dans la salle adjacente au niveau du mur opposé.
Représentation :

```
   A    B
1 [ ]--[ ]
   |
2 [ ]--[ ]
```

Dans cet exemple, en arrivant à droite (ouest) de A1, le personnage se retrouve à gauche (est) de B1. En allant au contraire vers le bas (sud) de A1, il se retrouvera en haut (nord) de A2.

Idéalement, le déplacement du personnage doit pouvoir être effectué par le joueur directement depuis le clavier (sans avoir besoin de valider manuellement un déplacement avec la touche entrée).

De plus, le joueur dispose, pour son personnage, des caractéristiques suivantes :
- Points de vie (maximum et actuels)
- Valeur d'attaque
- Valeur de défense
- Nombre de clés

Partie 2 - Moteur de combat

Lorsque le personnage cherche à entrer sur la case d'un adversaire, le personnage reste sur palce et à la place, un round de combat se passe. Cela se déroule selon les étapes suivantes :

- Le personnage frappe le monstre avec sa valeur d'attaque
- Le monstre réduit cette valeur d'attaque de sa propre valeur défensive, et perds ce nombre de points de vie
- Si le monstre est encore en vie, c'est à son tour d'attaquer avec sa valeur d'attaque
- Au personnage à présent d'utiliser sa valeur défensive et de perdre quelques points de vie.

A noter : il y a toujours au moins un point de vie de perdu, même si la valeur défensive est supérieure à la valeur offensive.

Exemple :

- Alice (10 pv, 3 attaque, 2 défense) attaque un Blob (5 pv, 1 attaque, 1 défense)
- A frappe B pour 3 dégâts
- B réduit de 1, donc subit 2 dégâts. Il lui reste 5-2 = 3 pv restants.
- B frappe A pour 1 dégât
- A réduit de 2 mais reste au minimum de 1 dégât. Il lui reste donc 10-1 = 9 pv restants.

Se déplacer vers un montre n'effectue qu'un seul round de combat. C'est au joueur de décider ensuite s'il souhaite faire un nouveau round ou aller voir ailleurs.
Le joueur à tout intérêt à minimiser la durée des combats : non seulement un ennemi qui est tué ne contre-attaque pas durant le round (économisant au minimum 1 pv au joueur), mais la victoire au combat donne également un bonus permanent de 1 point de valeur d'attaque et de 1 pv maximum. Ce bonus incite donc le joueur à affronter les ennemis les plus faibles en premier afin d'optimiser les rencontres suivantes.

Afin d'aider le joueur à effectuer les choix les plus pertinents, il faudra afficher les caractéristiques des ennemis du niveau, par exemple en listant tous les adversaires possible avec leurs statistiques, ou encore en affichant celles du plus proche.

Les monstres devant conserver leur nombre de points de vie lorsque le joueur se déplace d'une salle à une autre, il faudra trouver une manière de conserver en mémoire l'état des différents monstres du labyrinthe entier !

Partie 3 - Objets et inventaire

Lors de son déplacement, le personnage peut rencontrer différents objets dont voilà le fonctionnement :

- Une clé est ramassée et ajoutée à l'inventaire du joueur.
- Une porte consomme une clé et disparait ("s'ouvre"). Si aucune clé n'est possédée, la porte est infranchissable.
- Une potion redonne toute sa vie au personnage (ses pv deviennent égaux à leur maximum).
- Un powerup sont permanents et se déclinent en trois version : une qui donne un bonus de 1 point de valeur d'attaque, une qui donne un bonus de 1 point de valeur de défense et une qui donne un bonus de 3 points de vie maximum.

Les clés possédées par le joueur ainsi que les caractéritiques du personnage devront être affichées lors de la partie.

Partie 4 - Interfaces additionnelles

Le jeu devra afficher, à son lancement, un menu principal avec au minimum les choix suivants :

- Nouvelle partie
- Crédits
- Quitter

Nouvelle partie lancera le niveau nommé ’niveau 1’ (cf partie 5) en positionnant le personnage sur une des cases du milieu de la pièce.

Voilà les caractéritiques de début à utiliser pour le personnage :

- Points de vie maximum : 10
- Points de vie actuels : 10
- Valeur d’attaque : 2
- Valeur de défense : 1
- Nombre de clés : 0

Les crédits devront faire figurer, au minimum :

- Le nom et prénom des élèves membres du groupe
- La date de dernière modification de votre travail

Quitter termine simplement le programme (validation par l’utilisateur optionnelle).

Partie 5 - Fichiers de niveaux

Afin de pouvoir permettre la création et l'édition de niveaux, ces derniers seront enregistrés dans des fichiers séparés qui seront tous selon le même format.
Un fichier niveau comporte en premier l'ensemble des cases qui forment la pièce. La nomenclature est la suivante :

- Mur : #
- Porte : o
- Clé :!
- Potion : §
- Bonus d'attaque, défense, points de vie : 1, 2, 3 respectivement
- Monstres : A, B, C...
- Sortie de pièce : ?

Ensuite, quatre lignes dans l'ordre Est - Sud - Ouest - Nord donnent le nom du fichier lié à une éventuelle pièce reliée à un certain côté de la pièce.
Exemple :

```
Est :
Sud :
Ouest : sidezone1.level
Nord : niveau2.level
```

Enfin, chaque monstre est décrit à l'aide de ses caractéritiques dans l'ordre Pv, Force, Armure. Les monstres sont séparés d'une ligne vide.
Exemple :

```
A
Pv : 5
Force : 1
Armure : 1

B
Pv : 20
Force : 3
Armure : 2
```

Lors de la soutenance, un fichier suivant ces normes vous sera fourni afin de tester votre projet si votre groupe n'en dispose pas.

Partie bonus

Le contenu de cette partie n'est pas nécessaire pour avoir 20 (cf grille de notation en annexe), mais pour les groupes qui souhaitent aller plus loin, voilà différentes pistes qui seront considérées lors de la soutenance.

Save/Load :
Une partie impliquant de nombreux niveaux pouvant être assez longue, il serais judicieux de permettre à un joueur de sauvegarder sa partie pour la charger par la suite. Cela impliquera de créer un fichier (permanent) qui est conservé après la fermeture du jeu et qui pourra être utilisé pour revenir à la situation précédente lors de la prochaine execution.

Qualité de production :
Bien que le sujet soit tourné vers l'utilisation de la console et l'ascii-art, l'utilisation d'un moteur graphique permettra de genre le jeu bien plus agréable et plus clair à suivre. Il est demandé d'utiliser SDL afin de rester dans l'esprit du projet (par opposition, notamment, à GTK+).
De plus, le sujet d'évoque pas la création du contenu du jeu. La création d'au moins 3 niveaux non-triviaux liés entre eux sera considérée (non trivial voulant dire ici qu'il existe au moins un moyen de perdre et un moyen de gagner). La création de niveaux pouvant être parfois longue et complexe, et non liée à l'enseignement de ce module, un travail supplémentaire à ce qui à été décrit ne pourra pas être valorisé.

Options de debug :
Afin de permettre un test plus rapide et plus simple de votre projet, il est possible de considérer différentes options de triche/débug : modification des caractéristiques du personnage, vies infinies, etc. Ces options devront être impossible à utiliser pour un joueur non averti !

Annexes

Grille de notation

Chaque ligne correspond à un point de la note finale sauf si précisé autrement.

- Code : 16 points
  - Compilation sans erreurs (1 point) ni warnings (1 point). Code commenté et lisible (1 point).
  - Partie 1 - Monde et personnage
    - Existence des salles et du personnage
    - Déplacement
  - Partie 2 - Moteur de combat
    - Affichage des caractéristiques des adversaires
    - Round de combat
    - Fin de combat et récompenses
  - Partie 3 - Objets et inventaires
    - Affichage de l'inventaire et des caractérisitiques
    - Power-ups et potions
    - Interactions clé-porte
  - Partie 4 - Interfaces additionnelles
    - Lancement d'une nouvelle partie
    - Crédits/Quitter
  - Partie 5 - Fichiers de niveau
    - Lecture des fichiers niveau
    - Chargement des données en mémoire (2 points)
  - Partie bonus
    - Save/Load (1 point)
    - Qualité de production (Création de niveaux) (1 points)
    - Qualité de production (Ergonomie et graphismes) (2 points)
    - Options de debug (1 point)
- Présentation : 4 points