Université de Bordeaux
Sciences et Technologies

Master de Bioinformatique - Parcours Biologie Computationnelle

Année 2019-2020

Projet de programmation et d'algorithmique

La chasse au Wumpus

Alexandre Cornier, Abdelghani Neuhaus

Sommaire

Introduction

Hunt the Wumpus (ou "Chasse du Wumpus" en français) est l'un des premiers jeux informatiques. Dans sa version d'origine, le joueur donne ses instructions par ligne de commande. Ce jeu a été programmé par Gregory Yob en BASIC lorsqu'il était étudiant à l'université du Massachusetts à Dartmouth. Hunt the Wumpus fut publié pour la première fois dans le journal « Peoples Computer Company » en 1973, puis à nouveau en 1975 dans « Creative Computing », et enfin en 1979 dans le livre « MORE BASIC Computer Games ». Dans certaines versions ultérieures, dont celle sur TI-99/4A (sortie en 1981), une carte différente était générée à chaque partie.

Figure 1 : Logo initial du jeu

Ce jeu ancien nous vient du plus profond des montagnes de l'Asie centrale. Il servait d'épreuve d'initiation, non pas pour pas devenir un valeureux guerrier, mais pour révéler qui serait le plus fin stratège du village.

De nos jours, le Wumpus est très répandu dans la culture populaire et est notamment l'une des mascotte du logiciel de communication (du même type que Skype) Discord.

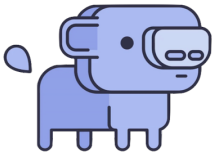

Figure 2 : Le Wumpus, figure de site Discord

Partie 1

Analyse du projet

1.1 Principes et règles du jeu

Le jeu consiste à vaincre le Wumpus, étrange habitant d'un labyrinthe, en le tuant avec une flèche. La grotte dans laquelle est caché le Wumpus est constituée de douze cavernes réunies par des couloirs. Le Wumpus dort dans l'une d'entre elle. La grotte abrite aussi deux chauve-souris qui peuvent déplacer le joueur dans n'importe quelle cave ; et deux puits : si le joueur tombe dedans, il meurt. Dans notre cas, nous avons décidé que chaque caverne serait reliée à trois autres caves. Le joueur disposera de trois flèches qu'il pourra utiliser pour abattre le Wumpus.

1.2 Contraintes à la réalisation

Les relectures du sujet ont soulevé plusieurs contraintes à gérer et à prendre en compte avant de commencer à concevoir le programme :

- Tout d'abord, il faut prendre en compte que le Wumpus connaît sa grotte parfaitement et qu'il n'est donc pas affecté par les actions de chauve-souris et la présence des puits ;
- Ensuite, il faut prendre en compte que le joueur entre dans une caverne vide pour commencer le jeu (sans Wumpus, ni chauve-souris, ni puits) afin qu'il y ait un minimum d'action dans le jeu ;
- Aussi, lorsque le joueur n'a plus de flèches, nous avons jugé que cela ne servait à rien de continuer la partie étant donné que le joueur ne peut plus gagner ;
- Également, lorsque le joueur tire sur une pièce adjacente à celle du Wumpus, cette action le réveille et il faut donc déplacer le Wumpus dans une de ses pièces adjacentes. Étant donné qu'il connaît sa grotte, il ne pourra pas aller vers un puits. Par contre, il peut être dans la même pièce qu'une chauve-souris ;
- Un autre problème soulevé est le fait que lors du lancement de la partie, il ne faut pas que certains personnages soient aux mêmes endroits (*par exemple : Wumpus et un des puits dans la grotte 1*)

Ces éléments ont donc été pris en compte avant de réaliser le jeu.

1.3 Affichage et lisibilité pour l'utilisateur

Le jeu sera affiché sur la console d'un terminal via la commande *"python3 script.py"*. Afin d'aider le joueur, une grotte sera affichée sur le terminal au début du jeu, mais aussi à chaque tour. Cette grotte affichera la position du joueur à l'aide de la lettre "J". Des instructions lui seront affichés :

— Position du joueur affichée à l'écrit mais aussi grâce à la grotte ;

— Message indiquant la présence éventuel du Wumpus, de puits ou de chauve-souris ;

— Message indiquant le nombre de flèches restantes avant et après l'action de tir ;

— Caves adjacentes à la position du joueur ;

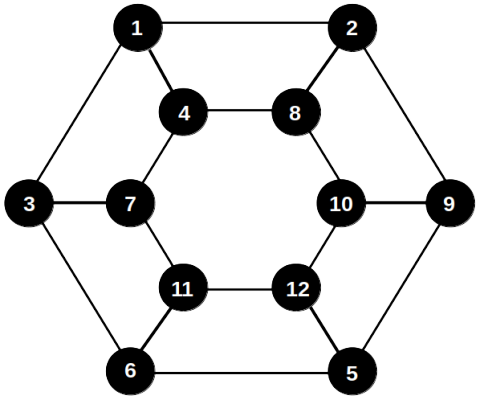

Figure 3 : Le labyrinthe du Wumpus

Partie 2

Conception du jeu

Dans cette partie, nous discuterons de comment nous avons agencé le jeu et ses différents constituants :

2.1 Initialisation du jeu et accueil du joueur

Au moment de l'initialisation du jeu et du lancement du jeu, le joueur est accueilli avec le nom du jeu "**WUMPUS**". C'est dès cet instant que le joueur débute son aventure. Pour rendre chaque utilisation unique et personnel, nous avons décidé de demander au joueur son prénom, ce qui permet de profiter pleinement de cette chasse au Wumpus.

Une fois l'initialisation de l'utilisateur réalisée, nous rentrons directement dans la phase de jeu en lui indiquant qu'il se trouve dans la grotte du Wumpus avec la phrase suivante "*Bob, vous êtes tombé(e) dans la grotte du Wumpus !*". Quand le joueur débute une partie, on lui affecte de manière aléatoire une caverne vide, par exemple *Bob, vous êtes actuellement dans la grotte 6* (Bob étant un exemple de prénom par défaut).

Ensuite, l'aventure commence pour l'utilisateur qui a toutes les cartes en mains pour réussir cette chasse. Pour une expérience de jeu plus confortable, nous avons créé une carte permettant au joueur d'avoir une meilleure représentation de la grotte. Cette carte est interactive, dans le sens où la position du joueur est représentée par le lettre "J" qui va se déplacer à chaque déplacement du joueur.

2.2 Création des éléments du jeu

a) Joueur

La création du joueur est l'élément le plus important dans ce projet. En effet, une interaction avec le joueur est essentielle pour le guider tout au long de sa partie, mais aussi pour son confort d'utilisation : une utilisation simple mais efficace permet à ce dernier de se déplacer et d'interagir avec son environnement, notamment par la capacité de tirer.

Nous avons voulu rendre chaque partie unique en faisant apparaître le joueur dans une des 12 chambres de manière aléatoire. Une fois cette phase d'initialisation terminée le joueur fait face à un premier dilemme qui est soit de parcourir la carte en se déplaçant de chambre en chambre ou bien de

tirer une flèche dans une des chambres voisines tout en sachant qu'il risque de réveiller le monstre qui dors au fond ce cette grotte s'il est proche de ce dernier.

b) Wumpus

Nous avons décidé d'initialiser le Wumpus comme celle du joueur, c'est à dire en lui attribuant une chambre de manière aléatoire avec tout de même une condition spécifique qui est de ne pas avoir la même chambre de départ que le joueur. Une fois cet étape réalisée, le Wumpus sommeil dans la grotte en attendant toute interaction avec le joueur.

c) Chauve-souris

Nous avons maintenant un plateau de jeu, notre personnage et un monstre qui dort paisiblement dans un labyrinthe. Pour compliquer cette chasse au Wumpus, nous avons décidé de ajouter des éléments qui changent la stratégie à adopter par le joueur pour pouvoir remporter la partie. En effet, l'introduction de deux chauves-souris, rend la tâche plus compliqué pour l'utilisateur.

Ces créatures sont elles aussi initialisées de manière aléatoire dans l'une des 12 chambres de la grotte mais ne peuvent pas apparaître sur une case identique à celle du joueur et du Wumpus. Leur rôle est d'augmenter la difficulté des prises de décisions par l'utilisateur car si ce dernier se déplace dans une chambre contenant l'une de ces créatures, celui-ci est immédiatement téléporter de manière aléatoire dans une autre chambre pouvant alors être celle du Wumpus et donc provoquer une fin de partie ;ou bien dans une chambre correspondant à un autre élément du jeu que l'on verra dans la partie suivante.

Pour conclure, même si les chauves-souris déplacent le joueur dans une autre chambre, en aucun cas sa position ne change tout long de la partie par un autre moyen.

d) Puits

Les puits permettent de renforcer le coté stratégique du jeu car il s'agit d'un danger mortel pour le joueur. Ils sont générés de manière aléatoire et ont une position fixe tout au long d'une partie. Ces derniers peuvent provoquer la mort du joueur de deux manières :

- Si le joueur choisit de se déplacer dans une case correspondant à celui d'un puits, c'est une fin de partie ;
- Si le joueur ce déplace dans une case correspondant à celle d'une chauve-souris, il y a alors deux chances sur 12 pour tomber sur une case correspondant à un puits et donc provoquer une fin de partie.

Ces éléments ont pour but, encore une fois, de compliquer la tâche de l'utilisateur dans la chasse au Wumpus.

e) Flèches

Pour atteindre l'objectif final d'abattre le Wumpus, nous mettons à disposition au joueur un arc avec 3 flèches. L'utilisateur pourra prendre la décision de tirer à n'importe qu'elle moment de la partie, à condition qu'il lui reste des flèches. Ce choix sera possible à chaque cycle de jeu par l'intermédiaire de cette phrase « *Que voulez-vous faire Bob ? Tirer (T) ou avancer (A) ?* ». A partir du moment où le joueur tire dans la case où se trouve le Wumpus, le joueur gagne la partie. Les flèches sont créées sous forme de compteur stockées dans la seule variable globale du jeu.

2.3 Éléments supplémentaires

Afin de personnaliser le jeu selon notre conception de celui-ci, nous avons pensé à ajouter une case flèche qui est générer de manière aléatoire, et permet donc au joueur de continuer la partie lorsque que son stock de flèches de départ est épuisé. Ainsi, ce dernier ne perd pas obligatoirement la partie lorsque son stock arrive à zéro ;la possibilité d'éliminer les chauves-souris, ce qui facilite la partie de joueur.

Partie 3

Réalisation

3.1 Stockage des données de positions

Les données de positions du joueur, du Wumpus, des chauves-souris et des puits sont stockées dans un même dictionnaire nommé *personnage*. Les différents constituants du jeu sont donc définis comme étant des clés auxquelles ont été associées au départ (comme expliqué précédemment) une valeur. Les positions du joueur et du Wumpus sont mises à jour lorsque le joueur se déplace (donc à chaque tour) et idem pour le Wumpus.

3.2 Initialisation

3.3 Boucle principale

La boucle principale est la dernière fonction créée. Elle consiste en l'appel successif des fonctions d'accueil et de créations des personnages. Il est ensuite défini exceptionnellement une variable globale *fleche*, de valeur 3 et qui est mise à jour uniquement lors de l'action de tir. Enfin, on répète les actions de choix et réalisation d'action (tir ou mouvement) et d'affichage de la localisation/statut du joueur.

3.4 Déplacement du joueur et du Wumpus

a) Joueur

Pour le déplacement du joueur, nous utilisons la liste contenant à chaque position i une liste de trois valeurs, qui contient les cases adjacentes à la position i. Par exemple, pour i=1, la liste en position 1 contient les cases adjacentes à la caverne 1 (donc 2, 3 et 4). Le joueur choisit lui-même vers quelle case se diriger et la valeur choisie devient la nouvelle valeur de la clé *'joueur'* dans le dictionnaire personnage.

b) Wumpus

3.5 Gestion des messages d'avertissements

La comparaison de la position du joueur avec la position des différents éléments a lieu après chaque déplacement effectué par le joueur. Si un des élément (puits, chauve-souris, Wumpus) est localisée dans l'une des cavernes que le joueur peut choisir, un message est affiché pour le prévenir. Il y'a également des messages d'avertissement lui indiquant le nombre de flèches qu'il lui reste (lorsqu'il ne lui en reste aucune, le joueur perd la partie).

3.6 Gestions des rencontres

Lorsque le joueur est face à une caverne contenant le Wumpus, une chauve-souris ou un puits, un message s'affiche pour le lui indiquer. Dans le cas du Wumpus, il est possible de le tuer afin de gagner le jeu. Il est aussi possible, pour le cas du Wumpus et des puits, d'arriver dans la caverne où ils sont et donc de perdre la partie. La rencontre avec une chauve-souris provoque un repositionnement aléatoire du joueur via la même méthode que lors de son initialisation.

3.7 Gestions des erreurs de saisie de l'utilisateur

Les gestions d'erreurs ont été principalement codées pour gérer les erreurs de saisies. Les différentes erreurs de saisies peuvent avoir lieu lors de la saisie du choix de l'action (tirer ou se déplacer) à effectuer par le joueur, lors de la saisie de la localisation de l'action. Nous avons défini une fonction **gestionErreurSaisie()** qui prend en argument la saisie effectuée et la compare à celle attendue par le programme pour fonctionner correctement. Aussi, pour faciliter les saisies du joueur lors de l'utilisation du jeu, nous avons intégré la méthode *.upper()* (ainsi, on peut saisir "A" en minuscule sans qu'un message d'erreur n'apparaisse.

Conclusion

Dans une version plus développée du jeu, nous aurions pu ajouter :

— La présence de flèche(s) à ramasser par le joueur au cours de la partie ;

— La possibilité de tuer les chauves-souris par l'action de tir et donc les supprimer du plateau de jeu ;

— La possibilité de choisir le nombre de pièces et de Wumpus ;

— Proposer une version graphique du jeu où le joueur voit les 3 grottes en face de lui.

Annexes

Dans une version plus développée du jeu, nous aurions pu ajouter :

- La présence de flèche(s) à ramasser par le joueur au cours de la partie ;
- La possibilité de tuer les chauves-souris par l'action de tir et donc les supprimer du plateau de jeu ;
- La possibilité de choisir le nombre de pièces et de Wumpus ;
- Proposer une version graphique du jeu où le joueur voit les 3 grottes en face de lui.