# Le labyrinthe de Willow

Thomas Hugard
Victor Trévinal

Novembre 2025

université de BORDEAUX

# 1 Analyse du problème

Le but du projet est de construire un jeu codé en Python3 dans lequel le joueur en interagissant avec la console, doit réussir à capturer un ennemi : le Willow. Le joueur et ce farouche ennemi se trouve tout les deux dans un labyrinthe hexagonal, composé de 12 salles. Afin d'attraper son ennemi, le joueur a à sa disposition 3 gâteaux, qu'il peut jeter dans n'importe quelle salle depuis sa position. Le but de cette manœuvre est de réveiller le Willow en l'appâtant avec l'odeur des sucreries, tout en se tenant dans une case adjacente afin de pouvoir le capturé. Pour se faire le joueur doit envoyer son gâteau dans la bonne salle où se trouve le Willow au risque de voir ce dernier se réveiller et possiblement bouger de salle. Le joueur sera guider grâce à des indications en fonction de son positionnement ; si il est proche du Willow ou d'autres éléments. En effet il n'y a pas que le Willow dans le labyrinthe, mais également des souris magiques qui peuvent téléporter le joueur à des cases aléatoires entraînant les conséquences qui vont avec. Il y a aussi des puits dans lequel le joueur pourra tomber par inadvertance. Si le joueur se retrouve dans la même que le Willow il mourra que ça soit de son initiative ou de part les souris.
Ainsi, à chaque tour le joueur pourra soit poser un gâteau, soit se déplacer d'une salle dans le labyrinthe.

# 2 Les solutions utilisées

Premièrement nous avons arbitrairement attribué des numéros aux salles. Comme nous avons prévu que le joueur se réfère à ces numéros nous lui avons donné la possibilité d'afficher une carte similaire à l'image ci-dessus dans le jeu.

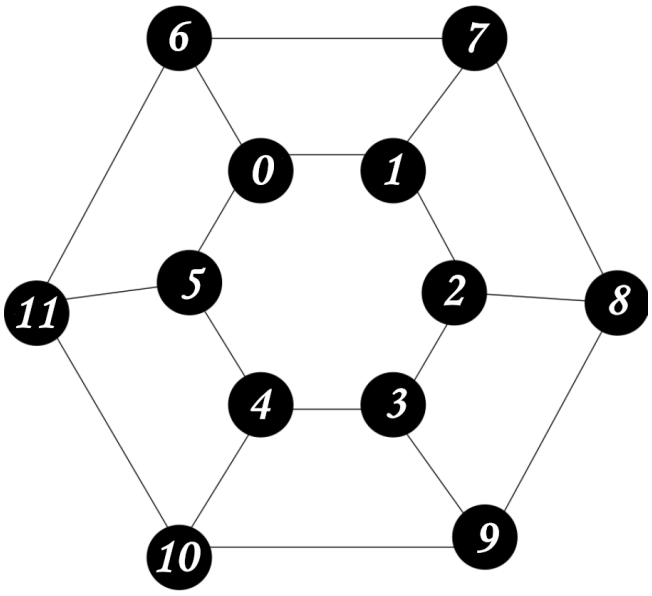


Figure 1: Attribution numéro de salle dans le Labyrinthe du Willow

Par la suite nous avons décider de choisir des listes pour contenir le numéro des salles appelée ici "l_salles" et une autre liste appelée "adjacent" pour le numéro des salles adjacents au placement du joueur.

Ensuite nous nous sommes dît que toutes les interactions du joueur et le déroulement des tours devrait passer par un menu principal, dans les faits on en a fait deux, chacun une fonction. Un premier pour lancer ou quitter le jeu, et le second pour le choix des actions du joueur à savoir : afficher la carte, se déplacer, jeter un gâteau ou retourner au menu principal.
Notre menu principal va donc invoquer les différentes fonctions de notre jeu. La première fonction qui se lance obligatoirement lorsque vous lancez une partie est une fonction d'initialisation, son but est de placer les deux puits, les deux souris, le joueur et le Willow, tout en s'assurant que le joueur n'est pas bloquer par les 2 puits et le Willow. Toutes ses positions seront déterminées aléatoirement l'une après l'autre, et vérifieront qu'elles ne chevauchent pas une des précédentes déjà placées. C'est l'endroit de notre code qui est sûrement le moins élégant, car nous utilisons des boucles while pour la vérification donc avec vraiment pas de chance, il peut avoir une grande complexité.

Si dans le menu de partie vous choisissez de vous déplacer, vous faites appel à la fonction deplacer(), qui a

pour but de vous demander dans quelle salle vous vous déplacez, en prenant en compte que la valeur retournée soit comprise dans les salles adjacentes.

A la fin de chaque action du joueur, 2 fonctions vont être lancées :
La première est pour déterminé les conséquences du déplacement du joueur, elle va vérifier la nouvelle position du joueur et la comparer à celle de tout les autres éléments et cela en 2 temps :

- Dans un premier temps toutes les conditions de défaites et victoires qui mettent donc fin à la partie
- Puis dans un second temps si dans les salles adjacentes il y a un élément et si oui renvoyer le message correspondant

Nous avons défini une fonction piege(), pour l'option du joueur de jeter des gâteaux dans une salle. Dans les consignes il n'était pas préciser comment et où le gâteau peut jeter un gâteau. Nous avons décidé qu'il pourrait jeter un gâteau sur n'importe quelle salle (depuis celle où il se trouve) à ses risques et périls. Dans le cas où le joueur veut jeter un gâteau la fonction va d'abord vérifier qu'il ne tape pas n'importe quoi comme nous avons pu le faire à d'autres endroits du code, le hic ici c'est qu'on aurait aimé avoir une façon de faire pour que si l'input échoue on bascule sur une commande de repli qui dirait qu'il y a eu une erreur. Pour l'heure le code crashera si le joueur tape n'importe quoi. Cependant nous avons tout de même réussis à faire en sortes que le joueur ne place pas deux gâteaux dans une salle tant qu'il y en a encore un dedans. C'est aussi la fonction gâteau qui gère le réveil du Willow suite au dépôt d'un gâteau et toutes les conséquences qui suive. Dans le cas ou le joueur s'est trompé de salle avec son jet de gâteau ; la décision du Willow de bouger ou de se rendormir se joue sur un pile ou face. Si il bouge la salle dans adjacente dans laquelle il ira est tiré aléatoirement.

La fonction afficherstat() quant à elle permet quand elle est appelée de rappeler au joueur ou il se trouve, les salles adjacentes à sa position et le nombre de gâteau qu'il lui reste. Nous avons fait en sorte de prendre en compte les différentes conditions pour le nombre de gâteau. Une conditions avec aucun gâteau dépensé, une autre avec 1 gâteau dépensé et la troisième correspondant a tout les gâteaux dépensés.

Nous avons aussi ajouté le fait d'afficher la carte a chaque demande du joueur. La fonction carte() comprend seulement des prints par ligne de carte qu'on voulait afficher. Dans les prints au niveau de la carte nous avons du ajouter des "raw string prefix" permettant de ne pas prendre en compte le "\" correspondant au diagonales de la carte.

Enfin nous avons ajouter la fonction regle(), qui permet d'afficher au joueur les règles du jeu. Cette fonction est constituée de deux dictionnaires, comprenant des paires de clés:valeurs, se réfèrents aux différentes règles du jeu. Le deuxième dictionnaire comprend un sous menu pour "les traquenards".

Dans l'ensemble du programme, nous avons ajouté des sleep du package time afin de "laisser respirer" le joueur. Avant lorsqu'on jouait au jeu avant les sleeps, toutes les informations et le déroulé du jeu apparaissait d'un coup dans la console, et le joueur devait même parfois scrollé en fonction de sa résolution, ce qui n'était pas très ergonomique.

# 3 Conclusion

Pour conclure, nous avons globalement réussit à faire ce que l'on voulait avec ce programme, nous pensons que les consignes sont respectées, et notre jeu tourne vraiment bien. En revanche si on pouvait l'améliorer aujourd'hui avec des connaissances que nous avons encore à acquérir, ça serait sur la gestion d'erreur.

Vous pouvez retrouver le programme du Labyrinthe de Willow sur le github :

https://github.com/EzRa2391/Willow