Programmation Web Avancée Côté Client
TP n° 6

Licence Informatique 2ème année
Université de Nice-Sophia Antipolis

Introduction

Dans cette séance, nous allons continuer le travail commencé lors de la dernière séance (TP n° 5), en ajoutant des nouvelles fonctionnalités et le traitement de conditions exceptionnelles.

1 Génération du labyrinthe

Tout d'abord, nous allons modifier l'algorithme de génération du labyrinthe pour le conformer aux règles du jeu *Labyrinthe*. Le plan du jeu est formé par des tuiles fixes et des tuiles mobiles qui sont disposées sur les cases d'un tableau de $n \times n$. Dans la version du jeu en commerce, $n = 7$, mais nous pouvons écrire du code plus général, qui marchera pour tout $n = 2i + 1$, $i = 1, 2, \ldots$.

Les tuiles fixes sont celles dont les coordonnées (i, j) sont telles que

$$i \pmod 2 = j \pmod 2 = 0.$$

Il y a donc $(\lfloor n/2 \rfloor + 1)^2$ tuiles fixes sur le plan du jeu. Le nombre total des tuiles est de $n^2 + 1$, parce qu'il doit y avoir une tuile en plus des cases du tableau : la tuile en trop sera posée par chaque joueur à tour à l'extérieur d'une rangée de tuiles et poussée dans le plan du jeu de manière à faire ressortir une nouvelle tuile du côté opposé. C'est ainsi que le labyrinthe changera pendant le jeu.

Les tuiles fixes sont disposées sur le plan du jeu selon une configuration prédéterminée comme suit :

- les quatre coins du plan contiennent des coins orientés avec leurs paroies donnant vers l'extérieur du plan ;
- les tuiles fixes occupant les cases des bords sont des croisements à T avec le côté fermé donnant vers l'extérieur du plan ;
- les tuiles fixes occupant les cases internes sont des croisements à T orientés comme les tuiles fixes du marge le plus proche ; si deux marges se trouvent à la même distance (c'est le case des tuiles fixes disposées sur les diagonales du tableau), c'est le marge qui se trouve dans le sens inverse des aiguilles d'une montre qui a la priorité : les tuiles fixes placées sur la diagonale qui va du centre au coin Nord-Ouest, par exemple, seront orientées comme les tuiles fixes du marge Ouest, celles sur la diagonale du centre au coin Nord-Est comme les tuiles du marge Nord, et ainsi de suite.

Les $n^2 - (\lfloor n/2 \rfloor + 1)^2 + 1$ tuiles mobiles sont disponibles dans des quantités prédéterminées comme suit :

– $4\lfloor n/2 \rfloor$ couloirs ;
– $2\lfloor n/2 \rfloor$ croisements à T ;
– $n^2 - (\lfloor n/2 \rfloor + 1)^2 - 6\lfloor n/2 \rfloor + 1$ coins.

Par exemple, si $n = 7$, on aura 12 couloirs, 6 croisements à T et 16 coins.

La configuration initiale du plan di jeu est générée aléatoirement en tirant à sort, sans remise, les tuiles mobiles à partir de cet ensemble, en les pivotant au hazard et en les plaçant sur la prochaine case libre jusqu'à remplir tout le tableau.

1. Écrivez un constructeur `Tuile` qui crée une tuile du type spécifié.
2. Remplacez le constructeur des cases du labyrinthe dans le code du TP n° 5 avec ce nouveau constructeur et réusinez le code pour utiliser ce constructeur à sa place.
3. Reécrivez la méthode qui génère aléatoirement le labyrinthe selon les règles expliquées ci-dessus.
4. Gérez la condition exceptionnelle où n est pair où $n < 2$.
5. Affichez la tuile en trop hors du tableau (à ce but, pensez à construir un petit tableau 3×3 pour la dessiner).

2 Pion

Nous représenterons les pions des joueurs (jusqu'à quatre) par les symboles `+`, `*`, `#` et `@`.

1. Ajoutez un morceau de code à la méthode qui génère le tableau pour placer sur le tableau le pion `+` dans une case aléatoire.
2. Modifiez la méthode d'affichage du labyrinthe pour toujours mettre en évidence (par exemple, en jaune) les cases accessibles par le pion `+`.

3 Mouvement du pion

1. Modifies la méthode de gestion des cliques sur les cases pour faire en sorte que, en cliquant sur une case accessible, le pion se déplace dans la case cliquée.
2. Gérez le cas où l'utilisateur clique une case non accessible à l'aide d'une exception et affichez un message d'erreur.