Projet de programmation

Julien Reichert Thomas Chatain

8 novembre 2013

Labyrinthe-démineur

Dans ce projet, vous aurez à concevoir un jeu qui utilise une bibliothèque graphique (`Graphics` ou `SDL`), fait des calculs sur un graphe, plus précisément une grille, et comprend une dose non négligeable d'interactivité, par des messages affichés et des saisies au clavier.

1 Principe

Le labyrinthe-démineur est un jeu dans lequel un personnage doit se déplacer d'un coin à l'autre dans une grille dont certaines cases sont piégées. Chaque case contient une information sur le nombre de mines parmi les quatre voisines, et le personnage dispose de cailloux qu'il peut jeter dans la direction de son choix en espérant activer une mine.

Votre travail sera de programmer ce jeu, en particulier la partie algorithmique consistera à répondre à quatre questions :

- Le personnage peut-il se frayer un chemin sans passer par la moindre mine (parcours en largeur) ?
- De combien de cailloux doit-il disposer au départ pour pouvoir faire exploser des mines et passer (Dijkstra) ?
- (bonus) Et si certaines cases peuvent contenir un certain nombre de cailloux, ramassables une seule fois ?
- (bonus) Quel est le nombre minimal de cailloux qu'il doit utiliser pour être sûr de survivre (donc ne jamais se déplacer sur une nouvelle case s'il est possible qu'elle soit minée) ?

2 Spécifications

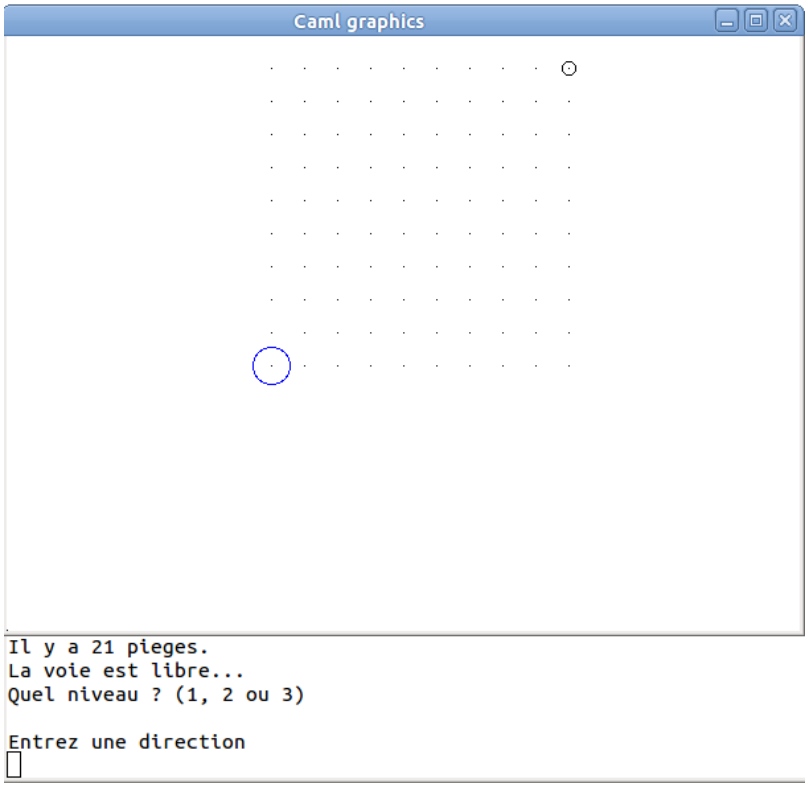
En entrée : Les dimensions de la grille et le nombre de cailloux au départ (sauf pour la dernière question). On pourra aussi demander à l'utilisateur combien il doit y avoir de mines (forcer un intervalle pertinent suivant la dimension), ou générer un nombre aléatoire (tout aussi pertinent). En sortie : La réponse aux trois questions, puis la partie doit être lancée si la réponse est oui. Si la réponse est non, soit générer une nouvelle grille jusqu'au oui, soit annoncer qu'un certain nombre de mines ont été retirées (choisir comment les retirer).

Possibilités annexes :
- Ajouter des murs, dont le nombre n'est pas annoncé sur chaque case. Soit le mur occupe une case (plus pratique), soit il se place entre deux cases (bon courage). Quand le personnage veut aller dans une direction où il y a un mur, un message le prévient. Les questions 2 et 3 doivent alors être adaptées pour tenir compte des deux paramètres.
- Limiter le nombre de déplacements du personnage avant qu'il ne perde par timeout (sauf s'il trouve des chronomètres...). Encore une fois, les questions 2 et 3 peuvent être adaptées.
- D'autres propriétés selon votre choix (exploser les murs, téléporteurs vers l'origine, etc.)

3 Affichage

Le labyrinthe sera affiché dans une fenêtre graphique (`Graphics` ou `SDL`), les messages seront imprimés sur le toplevel (de préférence sur la fenêtre si vous arrivez à le faire de manière esthétique) et les saisies effectuées soit en écrivant l'instruction dans le toplevel, soit (de préférence) en lisant la touche utilisée au moyen des fonctions du module idoine.

Voici une capture d'écran d'une version antique (2008) de ce jeu. Il est recommandable de faire mieux.



4 Évaluation

Vous écrirez un rapport de 4 pages maximum, décrivant le fonctionnement de votre programme, à envoyer par mail à `chatain@lsv.ens-cachan.fr` et `reichert@lsv.ens-cachan.fr`, avec le code source, au plus tard le **jeudi 12 décembre 2013**.

Les soutenances auront lieu le mardi 17 décembre 2013 en lieu et place des TP. Chacun de vous disposera de 20 minutes, questions comprises, pour présenter son travail et faire une démonstration de son code.