Rapport

Ce projet a été réalisé en collaboration avec Antoine CARREZ (15h), Elisa LEGRAND (8h) et Etienne LASHERME (7h).

Introduction :

Ce projet cherche à implémenter une version du jeu Igel Ärgern, un jeu de plateau où le but est de gagner une course de hérissons.
En plus des règles du jeu à implémenter, on a choisi trois fonctionnalités qui nous ont semblé intéressante (que l'on présentera dans la partie suivante). Le code proposé permet une utilisation simple avec un Makefile pour le compiler, avec la seule nécessité d'avoir gcc sur son ordinateur, toutes les options étant activables à partir d'un menu dans le jeu.

Fonctionnalités :

1. Murs :
   Le jeu de base ne propose que des "pièges" pour ralentir la course des hérissons. Nous avons décidé d'implémenter des murs qui se placent entre deux cases. La présence d'un mur interdit tout hérisson de passer d'une case à l'autre.
   De plus, pour donner plus d'intérêt à ces murs on a décidé de limiter la portée des pièges non pas à toute la ligne jusqu'au prochain mur.

   Pour implémenter ces murs, on a décidé de donner à chaque case un type :

   ```
   enum cell_type { BLANK, TRAP, WALL };
   ```

   Ainsi si une case est de type 'WALL', alors il y a un mur à sa gauche.
2. Bots :
   Pour pouvoir tester notre jeu et pouvoir jouer seul, nous avons décider de créer des bots. Ces derniers jouent aléatoirement, mais il permettent de tester les limites du jeu avec beaucoup de joueurs, de réaliser des paris à la manière de paris sportifs, ou de jouer avec plus de joueurs tout simplement.

   En plus de cette implémentation, nous avons dû rajouter la gestion des IAs dans le tour de jeu (elles jouent après les humains), dans l'affichage (en rajoutant des flèches pour montrer les mouvements effectués), ...
3. Modularité :
   Afin de permettre au joueur de profiter au maximum de toutes les fonctionnalités du jeu sans avoir à relancer le programme à chaque fois, nous avons créé un menu d'option. Ce

menu permet de modifier non seulement les paramètres de base (taille du plateau, nombre de hérissons par joueur), mais aussi de rajouter au choix les murs et les bots.
De plus, pour favoriser une rejouabilité nous avons décidé de placer aléatoirement les pièges et/ou les murs en fonction d'une proportion souhaitée (dans les options), tout en évitant les situations où la configuration du plateau est injouable (cas où toute une colonne serait un mur).

Ce choix de modularité étant de prime abord contre la suggestion du sujet de ne pas utiliser de malloc, nous avons décidé de majorer de façon logique toutes les dimensions.
En effet, les colonnes étant numérotés par des lettre, il y en a maximum 26. De même pour les lignes qui sont numérotées par des chiffres (maximum 9), et ainsi de suite...

Synthèse :

Nous avons réalisé un jeu qui fonctionne sans erreur. Il est prévu pour résister à toute entrée de l'utilisateur et de réagir correctement.
De plus, lancer des parties de 26 bots avec une cinquantaine de hérisson est amusant (attention à ne pas lancer des parties de 26 bots avec 300 hérissons sur de grands plateaux contenant des murs, la partie est trop longue).

Finalement nous avons essayé (et réussi selon nous) à rendre le code propre, lisible et compréhensible.

Nous avons également pensé à ces différentes extensions :

- Développer un serveur et un client pour jouer au jeu
- Utiliser ncurse pour améliorer l'interface et la jouabilité
- Ajouter de nouvelles cases ou des bonus récupérables