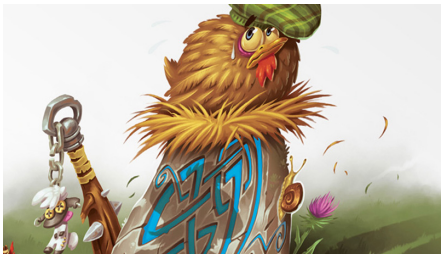
LO21 - Printemps 2023 - Projet Schotten-Totten



Dans ce projet, il s'agit de concevoir et développer une application permettant de jouer au jeu de société SCHOTTEN-TOTTEN, créé par Reiner Knizia. Pour les 20 ans de ce jeu populaire a été créé SCHOTTEN-TOTTEN-2 qui ajoute une dimension asymétrique au premier jeu. D'autre part, dans SCHOTTEN-TOTTEN ou SCHOTTEN-TOTTEN-2, il y a plusieurs variantes pour jouer (normal, tactique, expert).

Dans la suite du sujet, pour simplifier, quand on parlera d'une « *édition du jeu* », on parlera indifféremment d'une des éditions du jeu joué dans dans une des variantes.

Les différentes éditions utilisent les mêmes mécaniques de base mais diffèrent sur les cartes utilisées, le nombre de cartes distribuées, l'utilisation de cartes tactiques ou non, le nombre de bornes/murailles que l'on peut revendiquer à chaque tour, la présence ou non de certains éléments tactiques, *etc.*

Pour découvrir ces jeux, vous pouvez consulter par exemple les règles de SCHOTTEN-TOTTEN et celles de SCHOTTEN-TOTTEN-2, ou regarder des tutoriels (nombreux). Il est conseillé de jouer quelques parties (en commençant par la variante de base) pour bien comprendre les différents mécanismes du jeu.

1 Fonctionnalités attendues

L'application devra permettre de jouer des parties dans une édition/variante choisie. Il sera possible de jouer au moins à SCHOTTEN-TOTTEN dans la variante de base et la variante tactique.

2 Éléments d'interface

L'application doit permettre de paramétrer une partie :

- choisir l'édition/variante du jeu dans laquelle on souhaite jouer ;
- paramètres des joueurs (nom, joueur humain/IA, type d'IA si plusieurs, rôle selon édition) ;

L'application doit disposer d'au moins 1 type d'IA. Pour cela, vous pourrez développer une IA très simple qui prend ses décisions au hasard en fonction des possibilités.
L'application doit permettre de jouer la partie, de vérifier les revendications des joueurs, et de déterminer le gagnant.
Vous êtes libre d'organiser votre interface du moment qu'elle permet de piloter facilement l'application. Les visuels doivent seulement être suffisants pour comprendre l'état du jeu. Ainsi, pour les cartes, bornes ou murailles, il n'est pas spécialement demandé d'utiliser des graphismes. Soyez créatifs dans la façon de gérer les tours pour faire en sorte qu'un joueur ne puisse pas voir la main de l'autre.

3 Évolution de l'application

L'architecture de votre application devra permettre d'intégrer de nouvelles éditions/variantes du jeu sans remettre en cause le code existant (ou a minima). Les choix de conception devront donc permettre de rendre l'application évolutive et notamment de garantir la facilité d'ajout des composants suivants (sans impacter le reste du programme) :

— nouvelles IA joueurs,
— ajout de nouvelles éditions/variantes du jeu,
— éventuellement l'ajout d'éléments d'IHM de paramétrage liés aux nouveaux ajouts.

Vous démontrerez la pertinence de votre architecture dans votre rapport en expliquant comment ajouter ces éléments.

En choisissant une édition/variante du jeu que vous n'avez pas implémentée (et que vous n'avez pas à implémenter), à titre d'illustration, vous expliquerez dans votre rapport final comment faire (sans le coder) pour implémenter cette autre version (classes à créer, intégration dans l'architecture, éventuelles modifications de codes à prévoir, ...).

4 Consignes

— Le projet est à effectuer en groupe de 4 ou 5 étudiants (du même groupe de TD).
— Vous êtes libres de réutiliser et modifier les classes déjà élaborées en TD pour les adapter votre architecture.
— En plus des instructions standards du `C++/C++11/C++14`, vous pouvez utiliser l'ensemble des bibliothèques standards du `C++/C++11/C++14`.
— L'interface graphique est à réaliser en utilisant le framework Qt.

5 Livrables attendus

5.1 Rapports intermédiaires

Pour mener ce projet, vous devrez vous organiser de manière efficace. Il sera donc important de tenter de réfléchir aux tâches que vous allez devoir réaliser en vous posant pour chacune d'elles les questions suivantes :

— Cette tâche peut-elle être encore découpée en sous-tâches ?
— Quelle est la complexité de la tâche ?
— Quelle est la durée estimée pour réaliser cette tâche ?
— Quelles sont les relations de dépendance entre cette tâche et les autres ?

Vous devrez ensuite prioriser vos tâches (indispensable, importante, utile, bonus) et les répartir entre les différents membres du groupe de projet.

Au cours du projet, il vous est demandé de rendre compte de votre organisation. Vous devrez ainsi rendre 3 comptes rendus (10-15 pages maximum) lors des semaines suivantes :

1. la semaine du 27 mars ;
2. la semaine du 15 mai ;
3. la semaine du 5 juin.

Chaque rapport comportera :

— La liste des tâches mises à jour (qui seront des tâches a priori macros pour le rapport 1) avec l'estimation de leur durée en mettant en évidence les nouvelles tâches par rapport au précédent compte-rendu (rapports 2 et 3). Ces estimations et leurs mises à jour sont obligatoires.
— L'affectation (répartition a priori) des tâches (restantes) aux différents membres du groupe.
— Pour les comptes rendus 2 et 3, l'état d'avancement par rapport au précédent compte rendu : la répartition entre les membres du groupe et la durée a posteriori des tâches déjà effectuées, ou un pourcentage d'état d'avancement pour les tâches en cours avec la durée de travail déjà fournie par chacun des membres de groupe. Vous êtes particulièrement attendu sur ce retour qui devra être complet à chaque fois.

— Un bilan sur la cohésion de groupe et l'implication de chacun.

Bien que la liste des tâches s'affinera au cours du temps, vous devrez être précis sur les tâches à court et moyen-terme par rapport au rendu du rapport.

1. Le premier compte-rendu proposera aussi une analyse assez complète des différents concepts qui apparaissent dans le jeu et qui devraient a priori apparaître a minima dans votre architecture. Cette analyse se fera à partir des différentes éditions/variantes du jeu. À ce stade, il n'est pas exigé d'expliciter toutes les relations qui existent entre les différents modules. Il n'est pas exigé non plus de proposer un modèle qui intègre des éléments d'implémentation. Il s'agit avant tout d'un modèle conceptuel qui exige néanmoins d'avoir bien compris le jeu et ses constituants (il faut savoir jouer, il faut donc jouer !).
2. Le deuxième compte-rendu proposera aussi une première architecture assez complète qui intègre aussi des modules liés à l'implémentation. Il faudra reporter les associations principales ainsi que les prémisses des hiérarchies de classes existantes dans votre système. Cette architecture peut encore ne pas être définitive.
3. Le troisième compte-rendu rapportera l'architecture courante en indiquant les éventuelles modifications depuis le deuxième compte rendu. Il détaillera de manière complète le ou les modules qui permettent de jouer et les interactions que ce ou ces modules ont avec le reste de l'application. À ce stade du projet, il est attendu que la plupart des modules soient fonctionnels. Vous devez donc en faire la preuve. Il devrait normalement être possible de jouer en mode console.

Les rapports devront être déposés sur moodle dans la partie prévue à cet effet avant la fin de la semaine mentionnée (avant samedi 23h59 partout dans le monde).
Il sera tenu compte de la qualité de ces comptes-rendus dans la notation (le manque de précision dans les éléments demandés sera pénalisé).

5.2 Livrable final

Le livrable final est composé des éléments suivants :

— **Code source :** l'ensemble du code source du projet. Attention, **ne pas fournir d'excutable ou de fichier objet**.
— **Video de présentation avec commentaires audio** : une video de présentation dans laquelle vous filmerez et commenterez votre application afin de démontrer le bon fonctionnement de chaque fonctionnalité attendue (max 10 min, 99 Mo).
— **Rapport :** Un rapport en format `.pdf` (max 20-25 pages) composée des parties suivantes :
  — un bref résumé de ce que permet votre application (en précisant parmi les opérations attendues celles qui ont été implémentées et celles qui ne l'ont pas été) ;
  — la description précise de votre architecture en justifiant vos choix ;
  — une argumentation détaillée où vous montrez que votre architecture permet facilement des évolutions (voir explications ci-dessus). Il est attendu de présenter des diagrammes UML qui illustrent correctement (et au bon niveau) les sous-parties que vous décrivez.
  — une description détaillée du planning constaté de votre projet ;
  — une description détaillée de la contribution personnelle de chacun des membres du groupe sur les différents livrables (cette partie sera notamment utilisée pour la notation). Vous évaluerez en % la part de contribution de chaque membre sur l'ensemble. Chaque membre devra aussi reporter une évaluation du nombre d'heures de travail qu'il a consacré au projet.

  Pour ce rapport, vous pouvez réutiliser les parties pertinentes qui ont déjà été rédigées dans les compte-rendus.

L'ensemble des livrables est **à rendre avant le 18 juin 23h59 au plus tard (partout dans le monde)**. Les éléments du livrable devront être déposés sur moodle dans la partie prévue à cet effet.

6 Responsabilités des livrables

Bien que tous les membres du groupe doivent participer à chacun des livrables en partageant le travail de manière équitable et en fournissant tous les éléments dont ils ont la charge, un des membres du groupe sera désigné comme auteur responsable (au niveau du rendu) d'un des 4 rapports (comptes rendus intermédiaires et rapport final) et de la video. Chacun de ces rendus devra avoir un auteur responsable différent.

7 Évaluation

Le barême de l'évaluation du projet sur 20 est comme suit :

- **Couverture des fonctionnalités demandées au moins en mode console** : 6 points
- **Intégration dans une application GUI Qt** : 5 points
- **Choix de conception et architecture** : 5 points. En particulier sera évaluée la capacité de l'architecture à s'adapter aux changements.
- **Évaluation des livrables** : 4 points (video, code source, rapport, compte rendus intermédiaires, respect des consignes).

Remarque : une note inférieure ou égale à 10/20 au projet est éliminatoire pour l'obtention de l'UV.

8 Conseils

- Vous devriez commencer à sérieusement réfléchir au projet dès début mars, les premiers jours servant à découvrir le jeu et apprendre à jouer.
- Le temps de travail attendu est de l'ordre de 5 à 6h de travail en moyenne par semaine et par membre sur chacune des 8-10 semaines actives du projet. Il est donc important d'avancer régulièrement et de ne surtout pas attendre les 2-3 dernières semaines.
- Il est fortement recommandé d'utiliser un logiciel de gestion de versions afin de faciliter le travail collaboratif.
- Plusieurs TDs utilisent certains concepts communs au projet afin de commencer vous familiariser avec les différentes entités de l'application qui est à développer. On ne perdra pas de vue que les questions développées dans ces TDs ne constituent pas une architecture forcément pertinente pour le projet.
- La partie difficile du projet est la conception de votre architecture : c'est là-dessus qu'il faut concentrer vos efforts et passer le plus de temps au départ.
- Il est conseillé d'étudier au moins les design patterns suivants qui pourraient être utiles pour élaborer l'architecture de votre projet : decorator, factory, abstract factory, builder, bridge, composite, iterator, template method, adapter, visitor, strategy, facade, memento. En plus de vous donner des idées de conception, cette étude vous permettra de vous approprier les principaux modèles de conception. Attention, cela ne signifie pas qu'ils doivent forcément être utilisés.
- Pour la persistance des informations, vous êtes libres d'élaborer vos formats de fichier. Il est tout de même conseillé d'utiliser XML et d'utiliser les outils XML de Qt.
- Au lieu d'utiliser des fichiers, vous pouvez utiliser un SGBD comme SQLite.
- L'apparence de l'application ne sera pas prise en compte dans la notation. Soyez avant tout fonctionnels. Ca peut être très moche du moment que c'est jouable. En particulier le design des différents éléments visuels n'ont pas à être élaborés : il faut simplement que l'on puisse jouer.