LO21 - Printemps 2018 - Projet AutoCell

Dans ce projet, il s'agit de concevoir et développer l'application AutoCell, destinée à simuler des automates cellulaires à 1 ou 2 dimensions. Avant de commencer, nous vous invitons à découvrir le mondes très riche des automates cellulaires en consultant, par exemple, la page WikipédiA correspondante.

1 Description des fonctions principales

Dans ce document, sont présentées les spécifications du fonctionnement de l'application demandée. Vous pouvez les personnaliser à condition de ne pas diminuer la complexité du projet. Ces spécifications vous laissent volontairement des choix de conception, tant fonctionnels que conceptuels et technologiques. Quels que soient les choix et adaptations que vous ferez, vous prendrez garde de les exposer et de les justifier dans le rapport rendu avec le projet. Il peut manquer des spécifications. Dans ce cas, faites un choix en l'exposant clairement dans votre rapport.

1.1 Différents types d'automates cellulaires

De façon générale, un automate cellulaire est caractérisé par :

- une dimension (dans cette application, nous nous restreindrons aux automates à 1 et 2 dimensions),
- un ensemble fini d'états que peuvent prendre les cellules,
- un voisinage (*i.e.* l'ensemble des cellules *voisines* à prendre en compte dans le calcul de l'état suivant d'une cellule),
- une règle de transition qui définit le calcul de l'état suivant d'une cellule en fonction de son propre état et de l'état de ses voisins.

Il existe donc une infinité de types d'automates cellulaires différents. Votre simulateur ne pourra donc pas simuler tous les automates cellulaires possibles.
Initialement, l'application devra au moins permettre de simuler :

- les automates cellulaires les plus simples en 1 dimension ;
- le jeu de la vie de John Horton Conway.

1.2 Evolution de l'application

L'architecture de votre application devra permettre d'intégrer de nouveaux types d'automate cellulaire sans remettre en cause l'application.
Vous devrez illustrer cette possibilité en implémentant un autre type d'automate de votre choix.
Les choix de conception devront donc permettre de rendre l'application évolutive et notamment garantir la facilité d'ajout des composants suivants :

- ajout de nouveaux automates cellulaires ;
- ajout de nouveaux générateurs d'états pour un automate cellulaire ;
- modification de la partie IHM sans impacter le reste du programme.

1.3 Manipulation - Éléments d'interface

Le simulateur doit permettre de simuler un automate cellulaire, *i.e.* visionner dynamiquement l'évolution d'un automate de génération en génération. Ainsi les fonctionnalités élémentaires suivantes sont attendues :

- configurer un automate ;
- enregistrer/charger un automate ;
- définir la configuration initiale d'un automate ;

— enregistrer/charger une configuration ;
— lecture, arrêt, remise à zéro d'une simulation.

Pour cette dernière fonctionalité, vous pourrez alors remarquer que les représentations dynamiques utilisées classiquement pour les deux types d'automate qui doivent être implémentés sont différentes.
Vous êtes libre d'organiser votre interface du moment qu'elle permet piloter facilement l'application.

1.3.1 Éléments de configuration

Les éléments de configuration dépendent du type d'automate cellulaire.

— Pour les automates cellulaires à 1 dimension, il faut au moins être capable de :
  — définir le nombre de cases ;
  — définir le nombre d'états ;
  — définir la règle de transition ;
— Pour le jeu de la vie, il faut au moins être capable de :
  — définir les dimensions de la matrice ;
  — définir le nombre min de voisins vivants au temps t pour qu'une cellule soit vivante au temps $t+1$;
  — définir le nombre max de voisins vivants au temps t pour qu'une cellule soit vivante au temps $t+1$;
— Pour l'automate de votre choix, vous définirez vous-mêmes les éléments de configuration.

Pour chaque automate, il sera possible de choisir manuellement l'état initial de chaque cellule ou de générer les états automatiquement à l'aide d'un menu déroulant (génération aléatoire, symétrique, etc...)
Vous êtes libres d'ajouter d'autres éléments de configuration si ceux-ci vous paraissent pertinents.

1.3.2 Enregistrer/Charger la configuration d'un automate

L'ensemble des éléments de configuration d'un automate peut être sauvegardé/chargé. Vous êtes libres dans le choix du format de stockage de ces informations (csv, xml, bdd...).

1.3.3 Lecture d'un automate

Il peut y avoir plusieurs modes de lecture d'un automate :

— Mode lecture simple : Dans ce mode, les transitions entre les différents états de l'automate sont réalisées automatiquement après un pas de temps configurable (vitesse de lecture)
— Mode pas à pas : Dans ce mode, l'utilisateur déclenche manuellement le passage de l'état associé à l'instant t à l'état associé à l'instant t+1

1.4 Sauvegarde du contexte

Au démarrage de l'application, l'état de l'application, les paramétrages présents lors de la dernière exécution sont récupérés.

2 Livrable attendu

Le livrable est composé des éléments suivants :

— **Code source** : l'ensemble du code source du projet. Attention, ne pas fournir d'exécutable ou de fichier objet.
— **Documentation** : une documentation complète en `html` générée avec `Doxygen`.

- **Video de présentation avec commentaires audio** : une courte video de présentation dans laquelle vous filmerez et commenterez votre logiciel afin de démontrer le bon fonctionnement de chaque fonctionnalité attendue (environ 5-7 min). Pour réaliser cette video, vous pourrez vous servir des logiciels Jing (Windows, Mac OS), RecordMyDesktop (Linux). Ces logiciels sont mentionnés uniquement à titre d'exemple.
- **Rapport** : Un rapport en format `.pdf` composé de 3 parties :
  - la description du contexte (qu'est ce qu'un automate cellulaire ? définitions/explications des automates que permet de manipuler votre application, etc.)
  - la description de votre architecture ;
  - une argumentation détaillée où vous montrez que votre architecture permet facilement des évolutions. En particulier, vous illustrerez cette fonctionalité en montrant les différentes étapes d'ajout de code pour implémenter un nouveau type d'automate.

  Vous pouvez ajouter en annexe de ce rapport des instructions à destination de votre correcteur si nécessaire (présentation des livrables, instructions de compilation, ...). Ce rapport devra être concis et pas dépasser si possible 20 pages (annexes comprises).

L'ensemble des livrables est **à rendre pour le 16 juin 23h59 au plus tard**. Les éléments du livrable doivent être rassemblés dans une archive `.zip`. L'archive doit être envoyée par mail aux chargés de TD suivant votre séance :

- lundi matin : Antoine Jouglet (`antoine.jouglet@utc.fr`).
- lundi après-midi : Nicolas Cambier (`nicolas.cambier@utc.fr`).
- mardi matin : Thomas Furhmann (`thomas.fuhrmann@utc.fr`).
- mercredi matin : lotfi.zaouche@utc.fr (`lotfi.zaouche@utc.fr`).
- jeudi matin : Bastien Frémondière (`fremondiere@deltacad.fr`).
- jeudi après-midi : Mohamed Amine Mkadem (`mohamed-amine.mkadem@hds.utc.fr`).
- vendredi après-midi : Benoit Cantais (`benoit.cantais@utc.fr`).

3 Évaluation

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

- **Couverture des fonctionnalités demandées** : 6 points
- **Choix de conception et architecture** : 5 points. En particulier sera évaluée la capacité de l'architecture à s'adapter aux ajouts.
- **Evaluation des livrables** : 6 points (code source, documentation, vidéo, exemples de fichiers, rapport)
- **Respect des consignes sur les livrables** : 3 points (échéance, présence de l'ensemble des livrables et respect des consignes sur les livrables).

Remarque : il est rappellé qu'une note inférieure ou égale à 6/20 au projet est éliminatoire pour l'obtention de l'UV.

4 Consignes

- Le projet est à effectuer en trinôme (ou en binôme).
- Vous êtes libres de réutiliser et modifier les classes déjà élaborées en TD pour les adapter à votre architecture.
- En plus des instructions standards du `C++/C++11`, vous pouvez utiliser l'ensemble des bibliothèques standards du `C++/C++11` et le framework `Qt` à votre convenance.
- Il n'y a pas de contrainte concernant les éléments d'interface et d'ergonomie. Soyez créatifs. Il devrait y avoir autant d'interfaces différentes que de trinômes.

5 Conseils

— Plusieurs TDs utilisent le thème du projet afin de commencer à 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 pour le projet. Celle-ci devra être complètement retravaillée en tenant compte de l'ensemble des entités du sujet.

— 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.

— 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 moche.