M1 ALMA/ATAL
X7II030

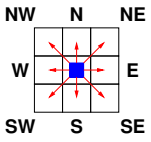
Université de Nantes
2014/2015

Projet de travaux pratiques
Automates cellulaires génériques

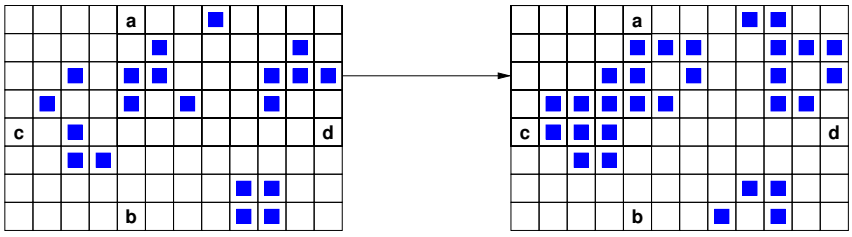
Contexte

Un automate cellulaire est composé d'une grille dont chaque cellule peut être vide ou contenir un pion d'un certain type, représenté graphiquement par un « *sprite* » adéquat. Une génération correspond à l'état de la grille à un moment donné. Pour passer d'une génération à la suivante, on considère chaque cellule et ses voisins (voir figure ci-dessous) et l'on applique une règle prédéfinie. Dans le cas particulier du *jeu de la vie*, il existe un seul type de pion et l'on utilise la règle suivante :

> *Si la cellule contient un pion qui a moins de deux pions voisins, il meurt de solitude ; s'il a plus de trois pions voisins, il meurt de surpopulation ; sinon, le pion survit à la génération suivante. Si la cellule est vide et qu'il y a plus de 2 pions voisins, un pion nait dedans à la génération suivante.*



Les 8 voisins d'une cellule



Génération n **Génération n+1**

Le jeu de la vie

La détermination de la nouvelle valeur pour chaque cellule se fait « *en parallèle* » sur toute la grille. Afin d'éviter les singularités sur les bords, on considère la grille comme un tore (les cases `a` et `b` sont voisines ; il en est de même pour `c` et `d`).

En fonction du nombre de types de pions autorisé et des règles de passage d'une génération à la suivante, il est possible d'utiliser les automates cellulaires pour modéliser de nombreux processus complexes : propagation d'un incendie de forêt en tenant compte de la répartition des différentes essences, dissémination d'un virus, ...

Travail demandé

On souhaite développer un logiciel de simulation d'automate cellulaire générique dont le comportement sera défini par des modules chargés dynamiquement à l'exécution (*plugins* [1]).

Fonctionnalités attendues

L'application devra au minimum offrir les fonctionnalités suivantes :
– Affichage graphique de la grille et de son animation au cours du temps ;

1. On pourra utiliser les fonctions `dlopen()`, `dlsym()`, ... pour charger un *plugin*.

– Définition au hasard de l'automate de départ en suivant des règles pré-établies (ex. : générer une grille de 500×300 cases avec au moins 30 % de résineux) ;
– Définition à l'aide d'une interface graphique de l'automate de départ (en utilisant les *sprites* proposés par le *plugin* courant) ;
– Chargement à l'exécution de *plugins* implémentant différents comportements (règles de propagation, types de cellules avec leur représentation graphique correspondante) ;
– Maîtrise des générations (point d'arrêt sur une configuration particulière, arrêt après n générations, ...).

On devra fournir au moins deux *plugins* bien différents (nombre de types de jetons possibles et comportement pour la mise à jour) avec le programme. La bibliothèque wxWidgets est installée localement au CIE dans l'arborescence `/comptes/goualard-f/local/{bin,include,lib}`. Cependant, le choix des bibliothèques utilisées pour réaliser le travail est laissé à l'appréciation des étudiants à la condition que l'application résultant soit compilable et exécutable au CIE. Les entorses à cette règle devront être exceptionnelles et seront à négocier *à l'avance* au cas par cas avec l'encadrant de travaux pratiques.

Notation

Les projets seront notés suivant plusieurs critères :
– Qualité du code écrit (modularité, commentaires pertinents, extensibilité, maintenabilité, ...) ;
– Qualité de de la documentation du code et des fichiers Doxygen associés ;
– Qualité de la documentation utilisateur (manuel fourni) ;
– Robustesse du code assurée par des assertions et des tests unitaires (utilisation de CppUnit) ;
– Déployabilité (utilisation correcte des *autotools*) ;
– Qualité du logiciel fourni : utilisabilité de l'interface, originalité des types de jetons proposés, fonctionnalités supplémentaires ...

Une plus grande attention sera apportée à la qualité du développement qu'à la complétion de l'application.

Organisation

Tous les projets devront être rendus sur madoc sous la forme d'une archive tarrée/gzippée[2] au plus tard le **lundi 15 décembre 2014 à 23h55**. L'archive déposée devra contenir l'intégralité du code source ainsi que le manuel au format LaTeX et PDF. Le langage de programmation de l'application sera C ou C++.

Le projet peut se décomposer en cinq grandes parties :
– Interface graphique de l'application ;
– Création des *plugins* et chargement dans l'application ;
– Simulation d'un automate en fonction du *plugin* courant ;
– Manuel d'utilisation (avec description des *plugins* fournis).

À l'intérieur d'un même groupe de travaux pratiques, un projet sera mené à bien par une *équipe* de quatre à six étudiants qui s'identifiera comme telle auprès de l'encadrant de travaux dirigés lors de la première séance dévolue au projet. La répartition des tâches au sein d'une équipe est laissée à l'appréciation de ses membres, l'encadrant se réservant le droit de vérifier à tout moment que chaque étudiant a bien une tâche attribuée et menée à bien.

Le projet de chaque équipe sera hébergé dans la Forge du CIE. L'utilisation correcte des facilités de gestion de version de la Forge fera partie de la notation.

2. Manipulation d'archives :
– Création de l'archive `titi.tar.gz` à partir du répertoire `titi` : `tar -vzcf titi.tar.gz titi/`
– Récupération du contenu de l'archive `titi.tar.gz` : `tar -vzxf titi.tar.gz`