Le jeu de la vie

Le T.P est inspiré du TP de Anabelle Colin, qui est disponible à l'adresse :
http://annabellecollin.perso.math.cnrs.fr/TPS/TP_3.pdf

Un Canevas QML est fourni pour vous permettre d'avoir une interface graphique, et de visualiser l'évolution sur la grille à chaque itération.

On s'attachera à ne pas avoir de code d'affichage pour l'objet GameLife.

La sauvegarde sera effectuée sous la forme d'un fichier correspondant à la QStringList de la fonction Wmanager::createStringFromPlateau(.....

L'ouverture d'un fichier correspondra à ce même format.

L'affichage sera géré uniquement par le Wmanager avec le QML.

(Le temps d'affichage entre deux itérations ne sera dans l'immédiat pas pris en compte)

Création contrôles

Ces contrôles ne seront pas immédiatement actifs mais permettront de définir l'aspect de l'interface finale.

Création d'une interface QML, qui outre le plateau, pourra avoir les boutons suivants dans un bandeau gauche par exemple :

- un bouton qui permettra de réinitialiser la grille

- un bouton qui lancera le nombre d'itérations indiqué par le combo

- un *TextField* (numérique) qui permettra de choisir le nombre d'itérations

- un bouton qui lancera une seule itération

- un bouton qui permettra de revenir en arrière une fois

- un bouton qui permettra de sauver la configuration en cours (dans un second temps un bouton qui permettra de choisir un fichier pour ouvrir un fichier sauvé)

- un *Slider* qui permettra de choisir un temps d'exécution plus ou moins rapide à l'affichage entre deux itérations.

- 2 saisies qui permettront de changer la taille de la grille

Création d'un objet GameLife suivant le prototype suivant :

```
class GameLife{

    public :
        GameLife();

        void initialisation();

        void load(std::vector <int> grid);
        std::vector <int> grid() const;

       // On utilisera : std::vector<std::vector<int>> neighbours;
       // pour les 8 déplacements {{-1, -1}, {-1,0}, etc....
        void nbAlivedNeighbours(int i, int j);
        void changeStatusOfCell(int i, int j);

        void play();
        void reversePlay();

        void setNewDimensions(int nbLignes, int nbColumns);

    private :
        int  m_nbLines;
        int  m_nbColumns;

        std::vector <int> m_grid;
        std::vector<std::vector <int>> m_history; // for reversePlay
};
```

Les règles du jeu sont décrites dans le document joint.
On pourra sauver les configurations décrites (clown, etc....)