

# TP 17bis Allocation dynamique, Composition et Agrégation

## Contexte et objectifs

L'objectif de ce TP est d'implémenter le « jeu de la vie<sup>1</sup> ».

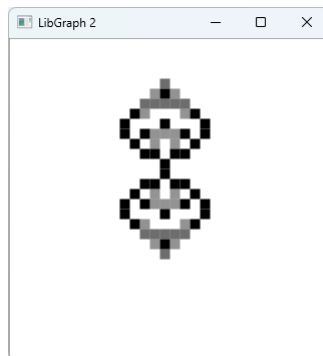


Figure 1 : Capture d'écran de l'application

## Exercice 0 Récupération des fichiers et présentation

- 📁 Récupérez le fichier *VEtudiant.zip*.
  - ➡ Ce fichier contient différents répertoires :
    - ➡ *Résultat* : une version compilée de ce que vous devez obtenir.
    - ➡ *doc* : une documentation au format html, disponible par le fichier *doc\html\index.html*.
    - ➡ *TP17bis* : Le répertoire du projet à compléter. Ouvrez la solution *TP17bis.sln*.
  - 📁 La solution Visual Studio est correctement configurée.
    - ➡ Compilez et exécutez, vous devriez obtenir une fenêtre blanche.
    - ➡ Comme vous pouvez le constater, l'application n'est évidemment pas fonctionnelle puisque la rendre fonctionnelle est l'objet de ce TP !

### 0.1 Présentation du code existant

#### 0.1.1 Architecture du programme

Le fichier *TP17bis.cpp* ne contient que la fonction **WinMain** qui gère les interactions de l'utilisateur via la bibliothèque *LibGraph2*. Après interprétation, les événements sont transférés à un objet de type **CGui** qui manipule l'objet principal des données du programme de type **CPetriDish**. C'est ce dernier qui modélise une boîte de Petri<sup>2</sup> contenant les différentes cellules de type **CCell** qui vont évoluer.

Chaque cellule a son propre état parmi « *naissante* », « *adulte* » ou « *mourante* ». Les cellules *mortes* sont tout simplement supprimées de la boîte de Petri. A la construction d'un objet **CCell**, la

<sup>1</sup> [https://fr.wikipedia.org/wiki/Jeu\\_de\\_la\\_vie](https://fr.wikipedia.org/wiki/Jeu_de_la_vie)

<sup>2</sup> [https://fr.wikipedia.org/wiki/Bo%C3%AEte\\_de\\_Petri](https://fr.wikipedia.org/wiki/Bo%C3%AEte_de_Petri)

cellule est automatiquement passée à l'état « *naissante* ». De cet état, elle peut soit devenir « *adulte* » puis « *mourante* », ou directement « *mourante* » par l'appel des fonctions **Adult()** et **Die()**.

Cet ensemble de classes et leurs interactions peuvent être représentés par le diagramme UML suivant :

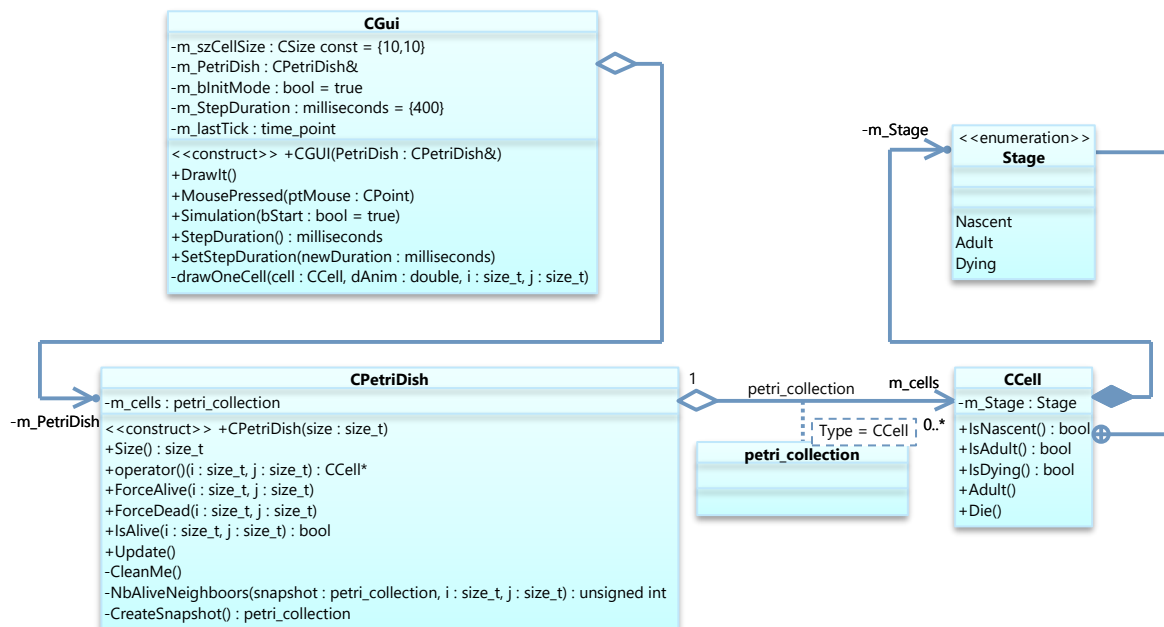


Figure 2 : Diagramme de classe de l'application

### 0.1.2 Interactions de l'utilisateur

Le programme propose deux modes de fonctionnement :

1. Le mode « initialisation », actif par défaut puis réactivable par l'appui de la touche « **I** »

Dans ce mode, l'utilisateur peut cliquer dans la fenêtre pour placer ou retirer des cellules dans la boîte de Petri.

2. Le mode « simulation » par l'appuie de la touche « **O** »

Dans ce mode, l'utilisateur ne peut plus cliquer dans la fenêtre et observe la simulation se dérouler. Il a tout de même la possibilité d'accélérer la simulation en appuyant sur la touche → ou ↑ et de la ralentir par les touches ← ou ↓.

## Exercice 1 Travail à réaliser



Le programme principal est déjà fait en grande partie. Vous n'avez le droit de modifier ce fichier qu'aux endroits indiqués par le commentaire **\todo**

### La classe CPetriDish











Commencez par définir correctement le type **petri\_collection** dans la classe **CPetriDish**.



Référez-vous au commentaire **\todo**.



Complétez le constructeur de la classe **CPetriDish**.

-  Complétez la fonction **CPetriDish::Size()**.
  -  Compilez et exécutez, la taille de la fenêtre blanche devrait être maintenant plus petite et carrée. Le code présent dans le programme principal crée une boîte de Petri devant être capable d'accueillir 32x32 cellules.
-  Complétez les fonctions **CPetriDish::ForceAlive()**, **CPetriDish::ForceDead()** et **CPetriDish::IsAlive()**.
-  Complétez la fonction **CGUI::DrawIt()** pour afficher toutes les cellules.
  -  Rappel, suivez l'instruction **\todo** de cette fonction.
  -  Compilez et exécutez, lors des clics dans la fenêtre, les cellules doivent apparaître et disparaître selon leur état.
-  Complétez maintenant les fonctions **CPetriDish::Update()**, **CPetriDish::CleanMe()**, **CPetriDish::NbAliveNeighbors()** et **CPetriDish::CreateSnapshot()**
  -  Compilez, exécutez et admirez ce qui se passe lorsque vous appuyez sur « O » après avoir initialisé quelques cellules à la souris.