Programmation concurrente
Game of Life

Claudio Sousa - David Gonzalez

11 décembre 2016

Image...

1 Introduction

1.1 Énoncé

2 Development

2.1 Architecture

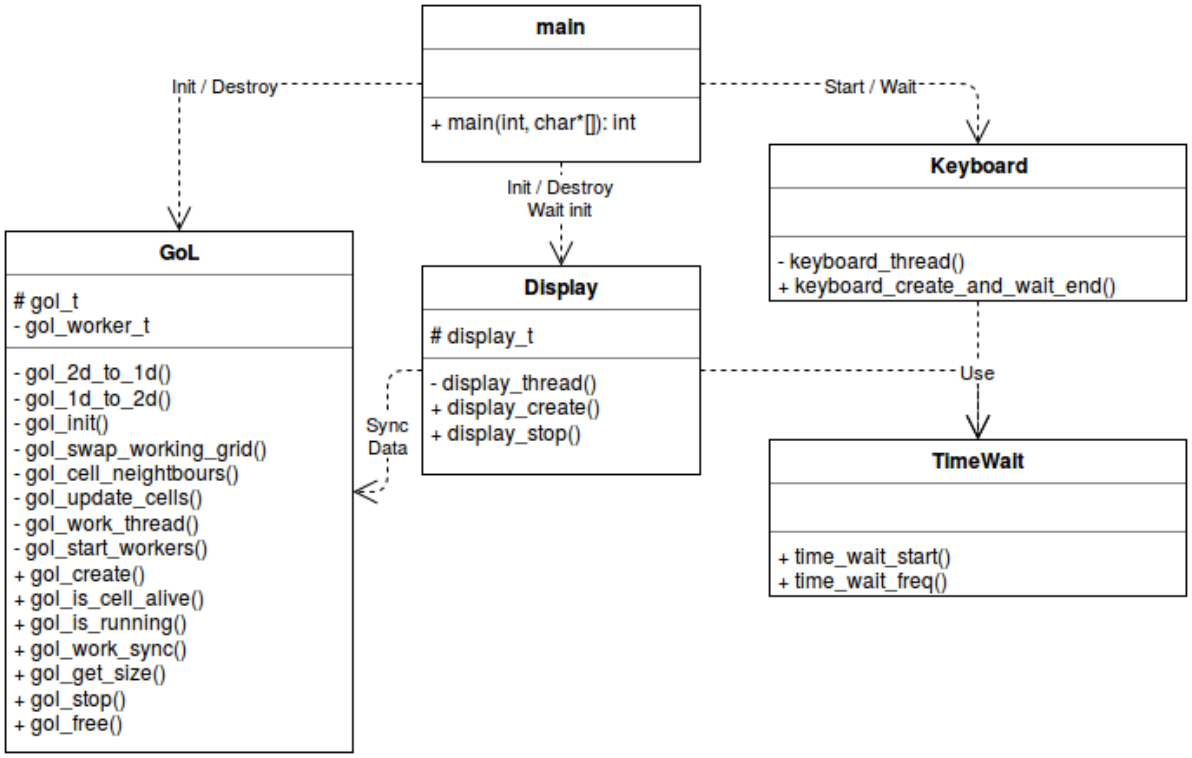

Figure 1: Architecture du Game of Life

2.2 Concurrence et parallélisme

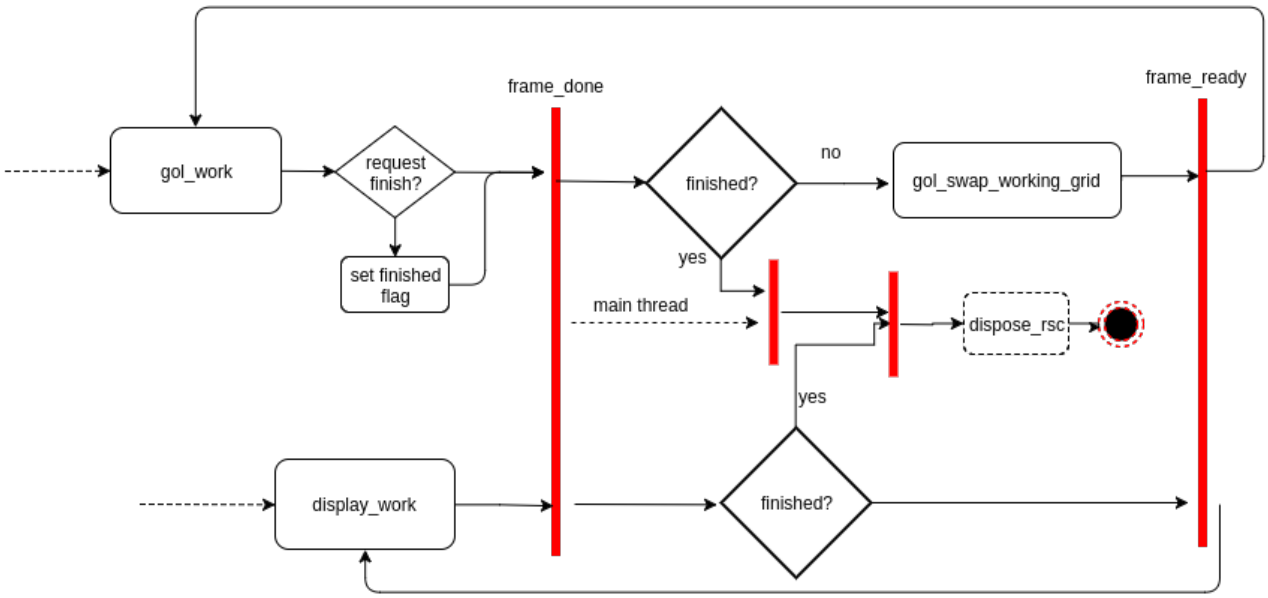

Figure 2: Synchronisation du traitement et de l'affichage

2.2.1 Synchronisation du traitement et de l'affichage

Les flèches à gauche du schéma symbolise les threads, n threads pour le traitement de la grille (haut) et le thread pour l'affichage de cette grille (en bas).

Tous les threads sont synchornisés à deux endroits, symbolisés par deux barres rouges verticales:

- *frame_done* synchronise tous les threads à la fin du traitement de l'état actuel de la grille.
- *frame_ready* s'assure que tous les threads attendent que la grille suivante soit prête à être traitée.

2.2.2 Condition de sortie

Lorsque la touche *ESC* est pressée, le thread du clavier se termine et libère le thread principal. Celui-ci met la variable *request_finish* à *true* et attend que tous les threads se termine.

Les threads du traitement de la grille vérifient cette variable avant la barrière *frame_done* et, le cas échéant, mettent une autre variable *finished* à *true*. Après *frame_done*, chaque thread vérifie l'état de cette variable et se termine s'elle est mise à *true*.

L'utilisaton de deux variables, et surtout la separation entre l'écriture et la lecture de *finish*, permet de s'assurer que sa valeur sera la même pour tous les threads entre *frame_done* et *frame_ready*.

La fin des threads redonne la main au thread principal qui libère les ressources du programme.

2.3 Méthodologie de travail

2.3.1 Répartition du travail