ÉCOLE POLYTECH'NICE SOPHIA
SI4
2019-2020

Julien Deantoni

Programmation multi paradigmes en CPP

TD numéro 4

Simulation à événements discrets

Objectif

L'objectif de cet exercice est de vous faire appliquer vos connaissance sur un example classique, la simulation à événements discrets (`https://fr.wikipedia.org/wiki/Simulation_%C3%A0_%C3%A9v%C3%A9nements_discrets`). Nous considérerons ici une version très simple d'un tel simulateur.
Habituellement, dans une simulation à événement discret, la simulation est faite d'une file d'événements temporisés[1], triés dans l'ordre temporel. la simulation est realisée en traitant le prochain événement dans la liste. Cet événement déclenche alors un comportement qui, potentiellement va ajouter des événements dans la file. La simulation s'arrête lorsque la file d'événément est vide après le traitement du dernier événement. Nous modifierons très légérement ceci afin de simplifier l'exercice.
Comme d'habitude, posez-vous la question du type des différents attributs et paramètres dans le système.

Énoncé du problème

Une Simulation possède un ensemble de Process enregistrés (confère figure 1). Elle spécifie le temps courant (i.e., le temps du dernier événement traité).
Un Process est associé à une et une seule Simulation. Il est lié à une fonction qui représente son comportement (le type `std::function` et plus précisément `std::function<void(unsigned int)>`, c'est à dire l'encapsulation d'un pointeur sur fonction prenant un entier non signé en paramètre (le courant de la simualtion) et renvoyant void, voir ici`https://fr.cppreference.com/w/cpp/utility/functional/function`). Un process connait la date de sa première exécution. Il possède aussi un attribut "period" qui spécifie la période à laquelle son traitement est effectué ou -1 si le process n'est pas périodique.
La classe Event connait le process qui sera exécuté lors de son traitement et le temps auquel le traitement devra être réalisé.

[1] i.e., un événement avec un temps, auquel il sera traité

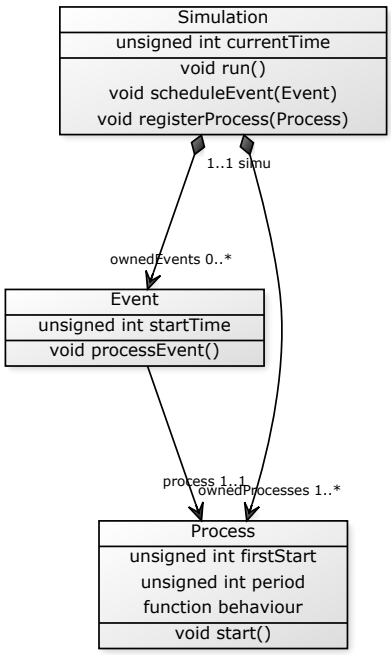

Figure 1: Structure de la simulation à événement discret.

Voici un example d'une utilisation possible de votre code:

```cpp
#include <iostream>
#include "Simulation.h"
#include "Process.h"
#include <math.h>

void f(unsigned int t){
  std::cout << "single exec ! " << std::endl;
}
void g(unsigned int t){
  std::cout << sin(t) << std::endl;
}
int main() {
  Simulation s;
  Process p1 = {s, f, 3};
  Process p2 = {s, g, 2, 1};
  s.registerProcess(p1);
  s.registerProcess(p2);
  s.run();
  return 0;
}
```

```cpp
#include <chrono>
#include <thread>

std::this_thread::sleep_for(std::chrono::seconds(aDuration));
```

Implémentez le système en prenant soin de réfléchir à vos choix de conceptions. Note: vous pouvez faire une version dont le temps de simulation correspond au temps de votre machine en ajoutant des "sleep". Pour ce faire,il faut ajouter ce genre de code: