Projet FSM

Raison :

J'ai choisi de faire la génération de code du langage C++ au langage C++, car le C++ est un langage objet proche de la machine. Une utilisation possible de ce projet serait par exemple dans le domaine du Serious Game où le C++ est très utilisé ainsi que les machines à états finis.

Fonctionnalités :

Les fonctionnalités supportées sont :

- l'état initial doit, ou être précisé dans la balise <scxml> sinon le premier état devient l'état initiale,
- l'état final peut être précisé par une balise <finale>,
- chaque état peut avoir plusieurs balise <onentry>, <onexit> et <transition>, qui peuvent toutes contenir la balise <raise>,
- <raise> peut contenir un attribut « event » qui va soir faire appel à une fonction renseignée par l'utilisateur associé à cet évènement soit soumettre l'évènement au graphe,
- une transition peut ne pas avoir d'évènement déclencheur précisé.

Limites :

Les limites de cette version sont :

- l'état initial n'est pas détecté par la balise <initial>,
- les actions de type « send » ne sont pas reconnus,
- les event nommés « q », en effet « q » est utilisé pour mettre un terme au programme,
- la distinction entre des transitions « external » et « internal ».

Tests :

Dans ce projet il y a 4 fichiers de test :

- graph1.scxml pour le test1 ***(faire "make test1" puis "./test1")***

  On test ici : - l'état initiale n'est pas précisé dans la balise <scxml>,
  - les transitions entre les états par événement.

- graph2.scxml pour le test2 ***(faire "make test2" puis "./test2")***

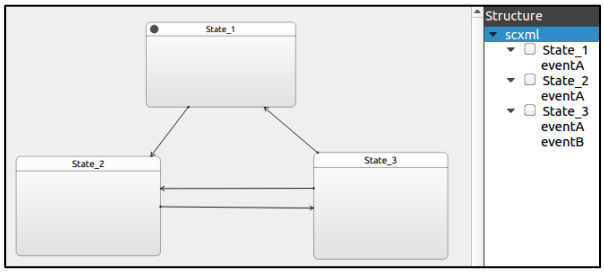
  On test ici : - un onentry et onexit pour un State,
  - un état final,
  - « raise » appelés par des transitions, par des onentry ou par des onexit, et qui soulève des fonctions comme doA(), doB(), etc...

- graph3.scxml pour le test3 ***(faire "make test3" puis "./test3")***

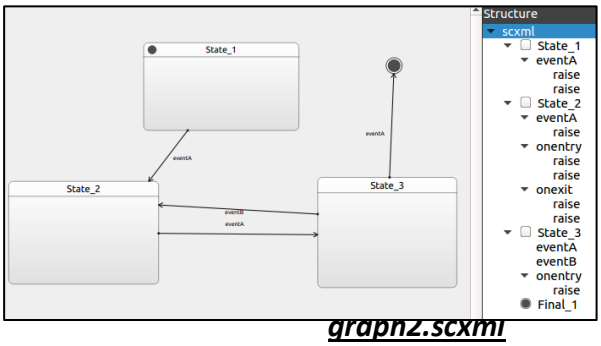
  On test ici : - plusieurs onentry et plusieurs onexit pour un State,
  - les transitions entre les états sans événements.

- graph4.scxml pour le test4 ***(faire "make test4" puis "./test4")***

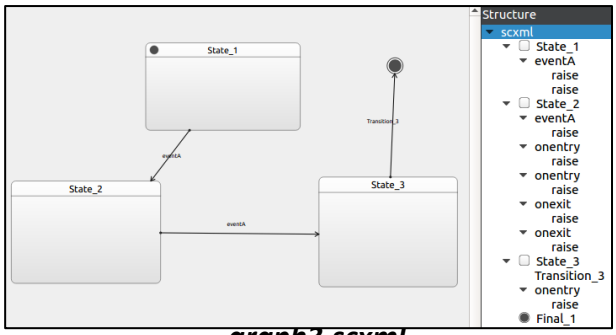
  On test ici : - « raise » peuvent soulever un évènement



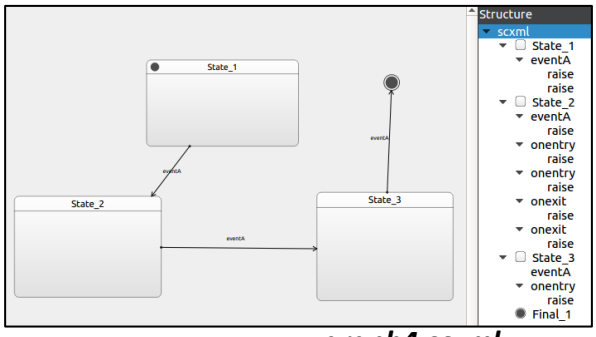
graph1.scxml



graph2.scxml



graph3.scxml



graph4.scxml

Mise en place du projet :

Prérequis : avoir une version de g++ à jour pour supporter l'option "–std=c++11"

Après avoir téléchargé le projet voici les commandes :

- pour compiler la bibliothèque :

  Exécuter les commandes suivantes dans le terminal

  "cd tinyxml"
  "make"
  "cp tinyxml.h ../tinyxml.h"
  "cp tinystr.h ../tinystr.h"
  "ar –q tinylib.a *.o"
  "cd .."

- pour compiler un des tests (ici N) du projet :

  "make testN"

- pour exécuter un des tests (ici N) du projet :

  "./testN"

Attention ! Entre chaque exécution ou compilation de différents testN, pensez à bien faire un *« make clean »*

ex : ***"make test1"***
"./test1"
"make clean"
"make test2"
"./test2"

Utilisation :

Prérequis : avoir réalisé la « Mise en place du projet » (chapitre ci-dessus) et connaitre les limites et fonctionnalités.

Deux choix s'offrent alors à vous :

- soit la version du main_test.cpp vous convient, vous pouvez alors changer un des fichiers graphN.scxml par votre scxml (si celui-ci respecte les limites et les fonctionnalités annoncées plus haut) en le renommant graphN.scxml. Puis recompiler et exécuter le projet.

"cp votre_fichier.scxml graphN.scxml"
"make testN"
"./testN"

ou

"./executable votre_fichier.scxml"
"g++ -o test500 main_test.cpp graph.cpp state.cpp transition.cpp action.cpp -std=c++11"
"./test500"

- soit la version du main_test.cpp ne vous convient pas, vous pouvez alors en créant un objet graph :
  - utiliser la méthode **« submit(string) »** pour soumettre un évènement au graphe,
  - utiliser la méthode **« registerGraph(string, pointeur sur fonction) »** pour renseigner une fonction associée à un évènement,
  - utiliser la méthode **« begin() »** pour lancer le début du graphe,
  - utiliser la méthode **« end() »** pour lancer la fin graphe.

  Enfin pour compiler et exécuter, suivez la première version en rajoutant vos sources.

Attention ! N'oublier pas de faire un *« #include "graph.hpp" »*