Programmation événementielle
Python, événements, callbacks,...

Manuel Munier

Version 14 octobre 2022 (12:02)

Objectifs

L'objectif de ce module est de comprendre le fonctionnement de la programmation par événements, c'est-à-dire de d'écrire un programme (en Python) capable de déclencher l'exécution de telle ou telle fonction *callback* pour réagir à la réception d'un événement.

Consignes du projet

- Le développement se fera en langage Python. Votre code devra impérativement être commenté.
- Vous devrez rédiger un court rapport fournissant la description de votre travail. Il ne s'agit pas de simplement fournir le code commenté ! À vous d'expliquer la logique de vos programmes et ce qu'apporte telle couche par rapport à la précédente. Bref, il vous faudra prendre un peu de recul...
- Une courte présentation ainsi qu'une démo seront exigées en fin de projet.

Travail à réaliser

À vous de choisir quelle application de test vous souhaitez développer. Mais faites en sorte que le cas d'étude que vous aurez retenu soit suffisamment "riche" pour exprimer toutes vos compétences...

1. interface graphique en Tkinter $\rightsquigarrow$ partie "dessin", c'est-à-dire création des widgets et des conteneurs, gestion des placements, etc.
2. interface graphique en TKinter $\rightsquigarrow$ partie événements, c'est-à-dire ajout de foncions callback à certains widgets, fonctions callback qui mettent à jour d'autres widgets, etc.
3. multithreading $\rightsquigarrow$ plusieurs threads (légers) capables de lire et/ou modifier des widgets $\rightsquigarrow$ notion de Timer, gestion des connexions réseau, etc.

Pour information voici quelques idées d'applications qui vous pourriez programmer :

1. Affichage du contenu d'une BdD SQLite (ex : tables `Fournisseur`, `Produit`, `Stock` vues en R207). Votre GUI en Tkinter doit permettre d'interagir avec les infos affichées : bouton de suppression d'une ligne, "formulaire" d'ajout de lignes, etc. Un peu comme en web dynamique (module R209), mais tout en Python à la place de HTML/PHP ☺
2. Application "Messenger like" : la GUI comporte principalement 2 zones : une pour saisir un message, une autre pour afficher la conversation. Chaque client se connecte au serveur. Quand un client envoie un message au serveur, celui-ci le rediffuse à tous les clients (qui l'affichent dans leur propre fil de conversation). Ce serait bien que les messages soient structurés (ex : JSON) pour que chaque client sache de qui vient le message ☺

Document rédigé en LaTeX sous Linux.