3A — STI6 — Projet d'Application — 2018

GUI Graphes/Automates Python3

Vincent HUGOT — vincent.hugot@insa-cvl.fr — Bureau CRI 09

24 avril 2018

Le but du projet est de réaliser un (ou plusieurs) prototype(s) d'une interface graphique pour la construction de graphes orientés et de diagrammes sagittaux d'automates d'états finis. Ceci est à réaliser en Python 3 ; le choix est ouvert pour la partie graphique, PyQt5 étant le choix par défaut.

Fonctionalités minimales
L'interface doit offrir la possibilité de placer des noeuds et des arcs – pouvant tous porter un texte – de les sélectionner, et de les déplacer / supprimer / renommer avec la souris et le clavier. *Bien entendu*, si on déplace un noeud, les arcs adjacents doivent se débrouiller pour rester connectés.

La structure logique du graphe (i.e. $G = (V, E)$) doit être mise à jour en harmonie avec la représentation graphique, afin que l'on puisse exporter les données du graphe sous différents formats. Pour l'instant, on se contentera d'un export au format texte.

De même, une structure "cartésienne" donnant la position (x, y) de chaque noeud doit aussi pouvoir être exportée.

S'il reste du temps
* Sélectionner et déplacer plusieurs noeuds simultanément, en conservant leur arrangement.
* Si plusieurs noeuds sont sélectionnés, possibilité de les réarranger automatiquement suivant différents motifs, e.g. en cercle, en grille,...
* Fonctionnalité "snap to grid"...

Qualité vs quantité
La finalité ultime de la chose étant de *déblayer le terrain pour un projet étudiant de plus longue haleine* – qui réutilisera peut-être votre code comme base – on visera moins à multiplier les fonctionnalités annexes qu'à obtenir un code robuste, clair, et dûment documenté, et une interface agréable et intuitive.

On vise donc à faire peu de choses, mais à bien les faire, et à documenter comment on les a faites.

Garder une trace documentée des échecs, des pièges, et des mauvaises idées est également intéressant.

Les sources utilisées (documentation tutorials etc) doivent impérativement être conservées.

Une grande partie du travail se résume donc à une recherche documentaire et à du prototypage rapide afin de trouver les façons les plus simples d'obtenir le résultat souhaité.

Si l'on trouve plusieurs façons viables de réaliser le travail (par exemple deux bibliothèques graphiques prometteuses) il est tout à fait possible d'écrire plusieurs versions du prototype (le temps le permettant) afin de comparer les approches sur pièces.

Suggestions d'outils graphiques

La principale difficulté ici par rapport à la partie graphique du projet Mon(s)tres est qu'on doit pouvoir interagir avec les éléments graphiques avec la souris, par exemple pour sélectionner et déplacer un noeud. Cela demande plus de travail que simplement dessiner l'automate.

* Le Widget Gaphas `https://gaphas.readthedocs.io/en/latest/` semble implémenter pas mal des fonctionalités requises. Il serait bon d'y jeter un oeil

* Les fonctionnalités Graphics View de Qt5, pour lesquelles il y a pas mal d'exemples (en C++) ici : `http://doc.qt.io/qt-5/examples-graphicsview.html` ; ceci bien sûr à adapter en Python (PyQt5).

* PyGooCanvas `https://wiki.gnome.org/Projects/PyGoocanvas` est une alternative à PyQt.

* TkInter ne semble pas avoir de support natif pour des fonctionalités interaction & "drag and drop" d'éléments graphiques, mais il est possible de les implémenter à la main avec des évènements `on_start on_drag on_drop`.