Programmation Avancée

# TD0 : Démarrage

Jacques-Henri Jourdan  Armaël Guéneau  Arnaud Golfouse

## 1 Installation des machines

Le but de ce TP est d'installer et configurer les outils que l'on utilisera dans les TPs à venir, pour programmer en OCaml et en Rust.

À fin de ce TP, **vous devez** avoir une machine configurée satisfaisant la checklist ci-dessous (machine personnelle ou compte informatique ENS). Nous fournissons une page d'aide à l'installation des outils :

`https://jhjourdan.gitlabpages.inria.fr/prog3-l3-ensps/install.html`

Vous êtes libres d'utiliser un éditeur de texte de votre choix (par exemple), tant que vous savez le configurer pour satisfaire la checklist. **Dans le doute, utilisez Visual Studio Code** et suivez les instructions correspondantes de la page d'aide.

Checklist pour OCaml :
— **opam** installé, avec une installation d'**OCaml 4.14**
— un éditeur configuré pour OCaml, avec : coloration syntaxique, indentation et **rapport d'erreurs en direct**
— le *build system* **dune** installé (via opam)
— toplevel amélioré **down** installé et activé

Checklist pour Rust :
— compilateur rust et cargo installés via **rustup**
— éditeur configuré pour Rust : coloration syntaxique, indentation, et **rapport d'erreurs en direct**

## 2 Application : du pixel-art distribué

Afin de continuer de tester votre installation tout en programmant un peu, on propose d'implémenter des petits programmes dessinant sur un tableau de pixels partagé, en envoyant des messages de dessin à un serveur commun [1].

1. `https://github.com/Armael/pixels`

Le vidéo-projecteur de la salle affiche le tableau de pixels partagé. On donne ci-dessous des liens vers des programmes minimaux de démarrage (à compléter) qui illustrent comment envoyer au serveur une commande pour le dessin d'un pixel coloré à une position donnée, qui sera alors affiché sur le tableau de pixels partagé. (Pour afficher plusieurs pixels, il suffit alors de répéter la commande avec différentes positions/couleurs.)

Code de démarrage OCaml : `pixels_ocaml.zip`
Code de démarrage Rust : `pixels-rust.zip`

Télécharger les codes de démarrage, les compléter et les exécuter pour vérifier que tout fonctionne bien (le tableau étant partagé, on changera la position du pixel pour une position de son choix bien identifiable). **Note :** penser à changer l'adresse IP du serveur pour l'IP donnée le jour du TP.

En profiter pour tester que la configuration de l'éditeur fonctionne bien : essayer de faire des erreurs (erreur de typage, erreur de syntax) et voir si elles sont bien reportées par l'éditeur au bon endroit.

Finalement, laisser libre cours à son imagination et écrire des programmes affichant des dessins plus complexes : motifs, texte, images statiques, ou encore dynamiques se déplaçant sur l'écran, etc.

# 3 Pour la prochaine séance

Afin de commencer à apprendre la programmation en Rust en douceur, on demandera de faire quelques **rustlings** (`https://github.com/rust-lang/rustlings`) chez soi, chaque semaine. Il s'agit de petits exercices très court, permettant de s'habituer à la syntaxe et aux constructions de base du language.

En particulier, on demande d'avoir fait les parties `00_intro`, `01_variables` et `02_functions` d'ici la semaine prochaine.

Pour faire les exercices, il suffit de se placer dans le dossier `rustlings`, puis de lancer la commande

```
rustlings watch
```

On pourra alors alterner entre son éditeur et le terminal, dans lequel s'affiche les erreurs concernant le test actuel.