# GIT, Maven et SceneBuilder

## 1 Introduction

Les TPs de JavaFX sont évalués de plusieurs façons :

— Un TP aléatoire sera évalué (1/8 de R2.02)
— Le projet Gribouille sera évalué (1/8 de R2.02)
— Un CTP final (1/2 de R2.02)
— La gestion GIT sera évaluée (1/3 de R2.03)

Aussi, vous devrez tout d'abord créer un projet sur la forge dans le projet regroupant votre TP :
`https://forge.info.unicaen.fr/projects/new?parent_id=but-r202-javafx-2025`.
Merci d'utiliser le nom "fx_" suivi de votre nom pour faciliter l'évaluation et d'ajouter vos encadrants de TP en tant que `manager`. Le seul module utilisé sera le dépôt de sources.

Chaque TP proposera deux parties :

1. une série d'exercices indépendants et guidés permettant d'apprendre
2. un exercice suivi plus autonome permettant de réutiliser ce que vous avez appris.
   N'hésitez pas à poursuivre cettre partie de TP en TP.

## 2 Apprentissage

La séance commence par quelques étapes en ligne de commande. Vous utiliserez ensuite votre IDE préféré.

### 2.1 GIT

Clonez votre dépôt git sur votre compte informatique. Ajoutez-lui le fichier `.gitignore` que vous trouverez sur eCampus. Attention au "." initial, que eCampus enlève souvent…
Vous pouvez ajouter à ce fichier des exclusions si votre IDE n'est pas Eclipse ou IntelliJ…

— Ajoutez ce fichier au dépôt avec la commande `git add .gitignore`.
— Vérifiez que le fichier est bien ajouté avec la commande `git status`.
— Commitez cette mise à jour avec la commande `git commit -m "ajout du gitIgnore"`.
— Mettez enfin à jour le serveur avec la commande `git push`.

La branche `master` ne contiendra **pas** d'autre fichier/commit.

— Créez une branche "TP1" avec la commande `git branch TP1`.
— Rentrez dans cette branche avec la commande `git checkout TP1`.

## 2.2 Maven

### 2.2.1 Archétype simple (ligne de commande)

Depuis la ligne de commande, entrez la commande `mvn archetype:generate`.

— Au bout de quelques secondes, vous devriez voir s'afficher une liste de $3000^+$ élements. Snif.

— Affinez la recherche en tapant "`openjfx:`". Attention, le ' :' est nécessaire car c'est une organisation !

— Choisissez enfin la ligne "`org.openjfx:javafx-archetype-simple`, dans sa dernière version.

— Entrez lez paramètres :

**groupId** iut.gon

**artifactId** test

**version** 1.0-SNAPSHOT (<entrée> suffit à valider la valeur par défaut)

**package** test

— validez la configuration. (On pourrait modifier le sdk ici, mais c'est pénible...)

Maven construit alors un dossier `test` avec son `pom.xml`, ses sources Java, aucune ressource, et un `module-info.java`. Modifiez le POM.xml pour changer la version de Java-FX de 13 à `21.0.2`. (cela évitera de télécharger plusieurs versions de la bibliothèque) Attention, la variable JAVA_HOME doit pointer sur un Java 17 ou plus...
(JavaFX 24 requiert un JAVA 22 ou + !)

Compilez le projet avec la commande `mvn compile`.

Testez le projet avec la commande `mvn javafx:run`. Vous devriez voir s'afficher une fenêtre avec le numéro de version de Java-FX et celui de votre JVM.

Ajoutez le dossier `test` à votre dépôt git et vérifiez avec un `git status`. Vous devriez voir ajoutés récursivement tous les fichiers du projet, à l'exception des fichiers générés (`.class`). N'oubliez pas de commiter.

### 2.2.2 Archétype fxml (IDE)

Ouvrez maintenant votre IDE.

Si vous utilisez Eclipse :

— utilisez comme *workspace* votre dossier GIT cloné à la question 2.1

— créez un projet Maven (ne cochez pas la case "skip archetype selection")

— filtrez sur openjfx

— sélectionnez l'archétype "`javafx-archetype-fxml`"

— entrez vos `GroupId`, `ArtifactId`, `Version` et `Package`

— modifiez la version de javafx en 21.0.2 (attention, Java 17 obligatoire)

— finalisez la création du projet.

Si vous utilisez IntelliJ :

— créez un projet JavaFX

— entrez son nom, son groupe et son artéfact.

— vérifiez bien qu'il sera créé dans votre dossier GIT, en Java et avec Maven.

— finalisez la création du projet.

Les deux projets sont quelque peu différents, mais ce n'est pas grave.

— Testez le (la commande "run" de votre IDE devrait fonctionner…

— Ouvrez un fichier `fxml` du dossier ressources.

— Ajoutez à la **VBox** un **ToggleButton** avec la propriété "`text`" réglée à "ON/OFF".
Vous aurez éventuellement besoin d'ajouter un import en tête de fichier. Inspirez vous de l'import de **Button**…

— Testez et appréciez la différence.

Pensez à ajouter votre nouveau projet au GIT et à le commiter. L'IDE devrait avoir détecté le dépôt, et vous n'avez rien à configurer de particulier (éventuellement ajouter le dépôt dans la perspective GIT d'Eclipse si vous vous en servez).

### 2.3 SceneBuilder

SceneBuilder est un éditeur visuel de fichier `fxml`. Vous devrez le configurer dans les préférences de votre IDE (Eclipse : Windows/Preferences/ puis General>Editors>FileAssociation où vous associez les fichiers ".fxml" à l'éditeur externe SceneBuilder, ou IntelliJ Files/Settings/Languages&Frameworks/JavaFX). Attention à ne pas cliquer sur le bouton "télécharger", car SceneBuilder n'est plus intégrable aux IDE depuis longtemps… Snif !
Vous pouvez aussi le lancer à la main et charger votre fichier FXML depuis ce dernier!

— Ouvrez votre fichier `fxml` avec SceneBuilder (bouton-droit puis "open with SceneBuilder").

— Ajoutez en tête de votre **VBox** une **Toolbar**

— Remplacez le contenu de la **Toolbar** avec 4 **ToggleButton**.

— Dans le panneau de droite, entrez le `toggle group` "groupe" pour 3 des 4 boutons.

— Cochez la case `selected` pour l'un d'entre eux.

— Testez votre fenêtre avec le menu "*Preview*". Quel est l'effet du groupe ?

— Sauvegardez et Quittez SceneBuilder

— Constatez les modifications du fichier `fxml` et testez.

— Commitez les modifications sur le dépôt GIT.

## 3 Réutilisation

Le projet Gribouille est un micro-logiciel de dessin. Vous en écrirez différentes parties au cours des 8 TPs de R2.02. Il est nécessaire de terminer chaque partie avant le TP suivant, qui ajoutera une nouvelle fonctionnalité.

— Revenez à la branche *master* et créez deux nouvelles branches "`gribouille_stable`" et "`gribouille_tp1`".
N'oubliez pas de rentrer dans la branche du tp1 de gribouille ; vérifiez avec un `git status` avant de continuer.

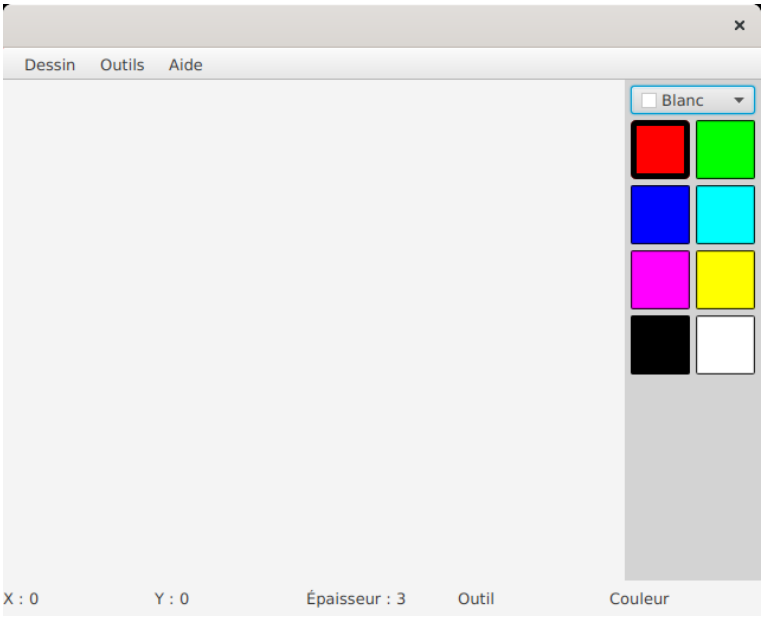
— Créez un projet Java-FX avec `fxml`. (Voir question 2.2.2)

— Supprimez les contrôleurs Java (gardez la classe avec le `main` !) et le(s) fichier(s) `fxml`.

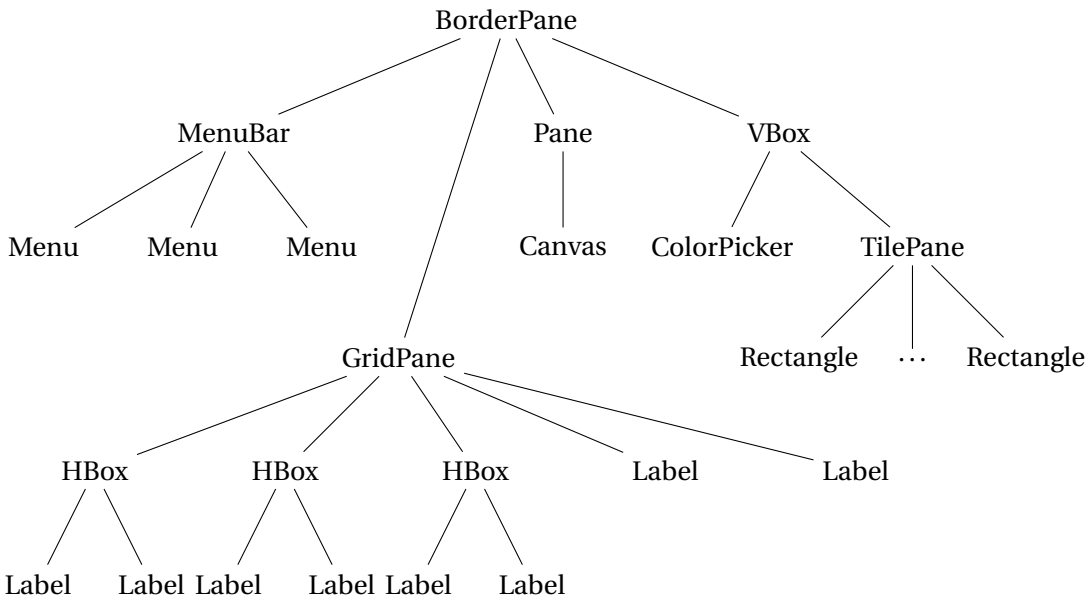
— Créez un nouveau fichier `CadreGribouille.fxml`. Sa balise principale sera un **BorderPane**.

— Ouvrez ce fichier avec SceneBuilder (ou modifiez le à la main…)
Le travail étant assez long, n'hésitez pas à tester et à commiter vos fichiers très souvent!

L'objectif est de réaliser la fenêtre suivante :

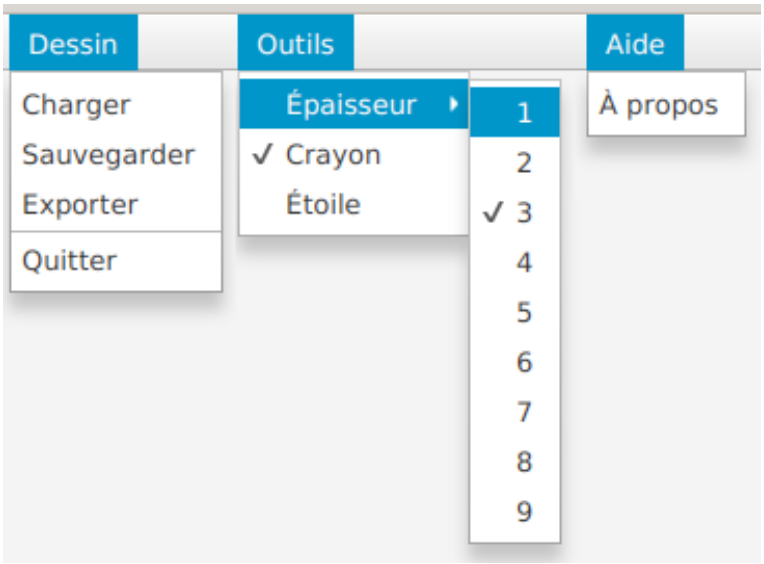



Voici un arbre présentant les composants à placer :




Ainsi, par exemple, la barre d'état (en bas du **BorderPane**) sera constituée d'un **GridPane** regroupant 3 paires de **Label** pour `x`, `y` et `epaisseur`, ainsi que 2 **Label** isolés pour `Couleur` et `Outil`.

La structure des menus n'est pas représentée dans l'arbre ci-dessus, et sera la suivante :




Il vous faudra donc 5 **MenuItem**, 1 **SeparatorMenuItem**, 1 **Menu**, 11 **RadioMenuItem** et 2 **ToggleGroup**.

Les 2 **ToggleGroup** permettent de grouper les **RadioMenuItem** “outils” et les **RadioMenuItem** “épaisseur” de façon à ce qu’un seul puisse être sélectionné à la fois.

## 4 Rendu

N’oubliez pas d’ajouter vos fichiers au GIT et de les commiter régulièrement.
N’oubliez pas le `git push` à la fin!

Une fois le travail terminé, revenez à la branche `gribouille_stable` et faites un `git merge gribouille_tp1`.
N’oubliez pas le *push* de cette nouvelle branche!