420-425-RK
Programmation Orientée-Objets :
Les classes et leurs relations
Travail Pratique no. 1

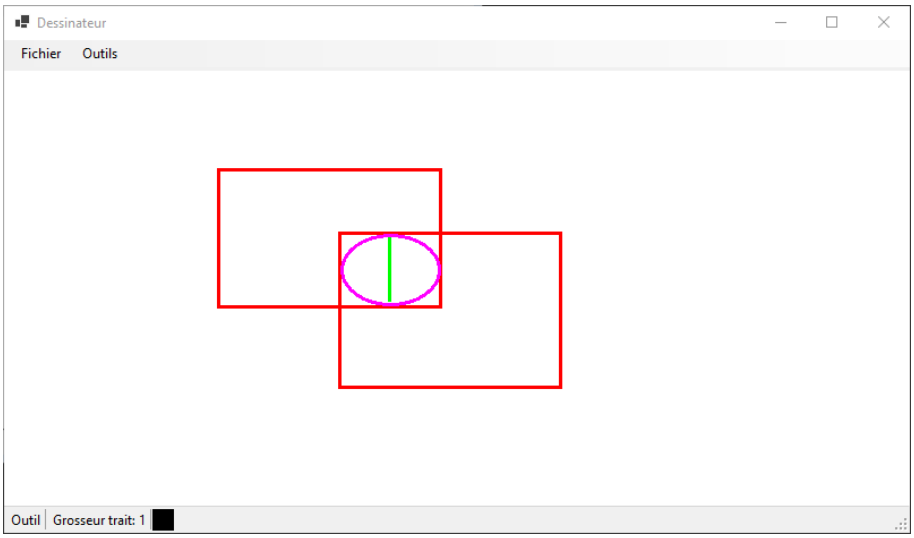
Éditeur graphique avec Windows Forms

Introduction

L'objectif de ce travail pratique est de programmer un éditeur graphique vectoriel qui permettra à l'utilisateur de dessiner certaines formes géométriques de base : des lignes, des rectangles et des ellipses. L'exercice vous permettra de vous familiariser avec des concepts que l'on retrouve régulièrement dans des librairies à caractère graphique.

Ultimement, votre éditeur graphique permettra à l'utilisateur de dessiner les formes énumérées précédemment à l'aide d'un ensemble de classes qui sont fournies par *Windows Forms*. Les classes en question seront énumérées plus loin dans ce document.

Voici un exemple de ce à quoi la fenêtre de votre éditeur graphique pourrait ressembler :



Cette fenêtre a comme composant principal une *PictureBox*. C'est sur cette dernière que l'on pourra réaliser les dessins. On peut également constater que la fenêtre comporte un menu et une barre de statut.

Le menu permettra à l'utilisateur d'enregistrer ou de charger un fichier préalablement enregistré, de sélectionner quel type de forme il désire dessiner, de modifier la couleur des formes dessinées, ainsi que l'épaisseur du trait utilisé pour dessiner les formes. Ce dernier item de menu devra faire en sorte qu'une nouvelle fenêtre sera affichée et demandera à l'utilisateur d'entrer au clavier l'épaisseur de trait qu'il désire utiliser.

Il est également à noter que votre programme devra offrir un menu contextuel à l'utilisateur lorsqu'il

effectue un clic-droit dans la surface du composant *PictureBox*. Ce menu contextuel devra comporter un item qui permet d'augmenter l'épaisseur de trait, un item qui permet de diminuer l'épaisseur du trait et un item qui fera apparaître la boîte de dialogue standard qui permet de choisir une couleur.

Finalement, tel que mentionné précédemment, votre fenêtre sera dotée d'une barre de statut afin de pouvoir indiquer à l'utilisateur le type de forme qui sera dessiné, l'épaisseur du trait utilisé et la couleur des lignes.

Détails techniques

Organisation des classes et abstractions

Il est attendu que votre programme devra exploiter au maximum les avantages liés au polymorphisme et aux injections de dépendances. L'application de la philosophie *Ordonner, pas demander ! – Tell, don't ask !* devra également primer, tout comme le principe de minimiser le nombre de responsabilités associés à une classe. **L'application de ces principes mènera donc à un code source dénué d'opérations de type *get* ou de propriétés C# (qui sont des *get* camouflés)**.

Vous devrez donc voir à mettre en place une architecture orientée-objet qui suivra ces orientations.

Classes de .NET à utiliser

Le composant *PictureBox* a été choisi pour la réalisation de ce travail car il permet d'obtenir une instance de la classe *Graphics* qui offre elle-même le nécessaire pour réaliser les dessins.

Toutes les classes nécessaires à la réalisation de dessin à l'écran proviennent de l'espace de nommage *System.Drawing*. Principalement, il est ici question des classes :

- PictureBox
- Graphics
- Pen
- Color (struct)
- Point (struct)

Il vous appartiendra, dans la phase d'analyse, de faire une bonne étude des pages de documentations mentionnées ci-dessus afin de déterminer comment procéder.

Mécanique de dessin

La mécanique à utiliser pour permettre de réaliser les dessins est somme toute assez simple. Par exemple, pour dessiner une ligne, on fera en sorte que l'utilisateur, après avoir choisi l'outil approprié, dessinera la ligne en cliquant une première fois dans la zone de dessin. Ceci indiquera le point d'origine de la ligne. Par la suite, on devra suivre les mouvements de la souris, et dessiner la ligne qui est déterminée par le

point d'origine désigné précédemment et la position actuelle de la souris, ce qui permettra à l'utilisateur de visualiser à quoi ressemblerait la ligne s'il effectuait un second clic à l'endroit où se trouve le curseur. Finalement, un second clic indiquera que la ligne dessinée à l'écran correspond bien à ce qu'il souhaite dessiner. La ligne ainsi terminée sera ajoutée, par exemple, à une liste dans la mémoire utilisée par votre programme.

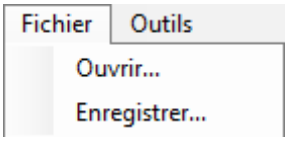
On peut envisager une mécanique similaire pour le dessin des rectangles et des ellipses.

Le composant d'interface graphique *PictureBox* émet une multitude d'événements qui pourront être interceptés afin d'arriver à notre fin.

Menus

La fenêtre de votre application comportera deux menus dans la barre de menu principale : un menu *Fichier* et un menu *Outils*.

Le menu *Fichier* aura l'apparence suivante :

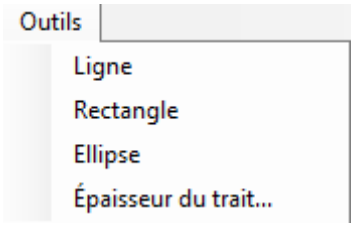


Lorsque l'utilisateur cliquera sur l'item "Ouvrir..."[1], une fenêtre de choix de fichier s'ouvrira. On remarquera qu'une telle fenêtre est standardisée dans l'univers de *Windows* et que *Windows Forms* l'offre sous forme de boîte de dialogue préfabriquée via la classe *OpenFileDialog*.

De façon similaire, lorsque l'utilisateur cliquera sur l'item "Enregistrer...", une fenêtre d'enregistrement s'ouvrira et permettra à l'utilisateur de choisir l'emplacement où le fichier devra être sauvegardé, ainsi que son nom. Cette fenêtre est également offerte, gracieuseté de *Windows Forms*, sous la classe *SaveFileDialog*.

Il vous revient de déterminer le format des fichiers texte que vous utiliserez pour sauvegarder les données de votre application. On pourrait par exemple penser à y placer les données de chaque objet dessiné à l'écran sur une ligne du fichier, en utilisant un caractère "spécial" pour séparer les données. Vous devez absolument supporter la sauvegarde et le chargement dans un format texte de votre choix. Pour aller plus loin (pas de points bonus) : supporter l'exportation vers un format d'image comme PNG, BMP ou JPG... la PictureBox permet-elle de.

Le menu *Outils* comportera 4 éléments, comme on peut le voir sur la figure suivante :

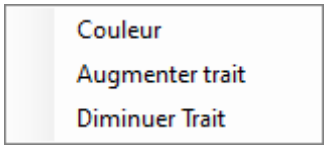


[1] Il est à noter que les points de suspensions suivent une longue tradition qui veut qu'ils indiquent qu'une autre fenêtre s'ouvrira afin de pouvoir donner d'autres informations

L'item *Ligne* fera en sorte que la prochaine forme dessinée par l'utilisateur sera une ligne. Ce faisant, l'indicateur du type d'outil sur la barre de statut sera mis à jour pour refléter cette réalité. Il en va de même pour l'item *Rectangle* et l'item *Ellipse*.

Un clic de l'utilisateur sur l'item *Épaisseur du trait...* fera apparaître une fenêtre à l'écran. Cette fenêtre contiendra un champ texte (*TextBox*) accompagné d'une étiquette (*Label*) appropriée. Elle comportera également deux boutons : un bouton portant l'étiquette "OK" et un bouton portant l'étiquette "Annuler". Le champ d'édition sera utilisé pour récupérer la largeur de trait à utiliser pour dessiner la prochaine forme géométrique.

Finalement, lorsque l'utilisateur effectuera un clic-droit dans la zone de dessin, votre application devra afficher un menu contextuel, comme le montre la figure qui suit :



Lorsque l'utilisateur cliquera sur l'item *Couleur...*, une boîte de dialogue de choix de couleur sera affichée à l'écran, afin d'indiquer la couleur à utiliser pour dessiner la prochaine forme géométrique. Lorsque l'utilisateur aura effectué son choix, l'indicateur de la barre de statut correspondant devra être mis à jour, afin de montrer la couleur choisie.

Un clic de l'utilisateur sur l'item *Augmenter trait* fera en sorte que l'on augmentera de 1 pixel la largeur présentement utilisée, alors qu'un clic sur *Diminuer trait* fera en sorte que l'on diminuera de 1 pixel la largeur présentement utilisée. Dans un cas comme dans l'autre, on verra à mettre à jour l'indicateur de la barre de statut qui montre la largeur de trait courante.

Vous à produire le programme qui permettra à l'utilisateur de dessiner des lignes, des rectangles, des ellipses à l'écran tel que décrit dans ce document.

Remise

Remise sur GitHub à la date et heure de remise finale.