PROJET : une histoire de tortue

LOGO [1] est un langage de programmation mis au point dans les années 60 par le mathématicien Seymourt Papert [2] et son équipe au MIT pour apprendre des rudiments de programmation à des enfants.

Dans la version classique, une petite tortue est dessinée sur l'écran de l'ordinateur et lorsqu'elle se déplace, elle laisse une trace derrière elle. Elle peut être commandée avec des instructions simples : `avance` d'un certain nombre de pas, `recule` d'un certain nombre de pas, `tourne à gauche` d'un certain angle, `tourne à droite` d'un certain angle. Lorsque les enfants maîtrisent bien son fonctionnement, ils peuvent obtenir de très jolis dessins.

Nous allons ici programmer une version un peu différente.

Vous trouverez ci-dessous la description du noyau fonctionnel et de l'IHM de l'application de mes « rêves », puis des précisions sur le travail attendu. Concrètement, vous n'aurez pas toutes les fonctionnalités à implémenter mais les choix de conception que vous ferez doivent permettre d'étendre votre application pour obtenir l'application idéale sans tout démonter et tout recommencer.

1 Noyau fonctionnel

Une tortue occupe forcément un point à coordonnées entières.

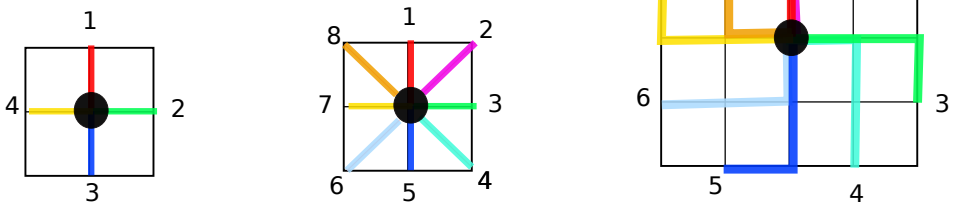
Contrairement à la version classique, notre tortue n'avance pas d'un certain nombre de pas ou ne tourne pas d'un certain angle, mais avance selon un ensemble de motifs donné. Ces motifs sont "ordonnés" et lorsqu'elle "tourne", elle passe, en réalité, d'un motif à un autre. Comme un dessin vaut mieux qu'un long discours voici quelques exemples d'ensembles de motifs (Figure 1).

Un motif n'est rien d'autre qu'un ensemble de déplacements. Ainsi le motif 1 de la figure 1a correspond à un déplacement vertical de 1 vers le haut. Le motif 1 de la figure 1c correspond à un déplacement vertical de 2 vers le haut suivi d'un déplacement horizontal de 1 vers la droite. Un déplacement est classiquement représenté par un vecteur : on peut donc représenter un motif par une liste de vecteurs.

Une tortue peut effectuer les instructions suivantes :

- GO : la tortue avance en suivant le motif courant ;
- GO k où k est un entier : la tortue répète k fois GO ;
- TURN : la tortue passe au motif suivant ;
- TURN k : la tortue répète k fois TURN, autrement dit passe au $k^{ème}$ motif suivant ;
- DRAW on (resp. DRAW off) : la tortue passe en mode dessin (resp. la tortue quitte le mode dessin) ; ainsi une tortue peut avancer sans dessiner (ce qui permet de réaliser des dessins non connexes, par exemple deux segments déconnectés).
- COLOR col, où col est une description de couleur, permet de changer la couleur de dessin de la tortue ; chaque motif dessiné par la tortue peut avoir une couleur différente et on doit pouvoir changer la couleur de n'importe quel bout du dessin correspondant à un motif (cf Figure 2).
- UNDO : annule la dernière instruction (GO, TURN, COLOR, DRAW) effectuée par la tortue ; on doit pouvoir annuler successivement toutes les actions réalisées depuis l'initialisation (ou la

1. `http://el.media.mit.edu/logo-foundation/`
2. `http://web.media.mit.edu/~papert/`



(a) La tortue peut avancer selon 4 motifs différents qui correspondent chacun à un pas dans une direction nord (1), est (2), sud (3) et ouest (4)

(b) La tortue peut avancer selon les 4 motifs précédents mais également en diagonale.

(c) Les 8 déplacements de cette tortue sont un tantinet plus complexe : on reconnait ceux du cavalier du jeu d'échec. À noter que les motifs 1 et 2 partagent un segment de même que les motifs 3 et 4, 5 et 6, 7 et 8.

FIGURE 1 – Chacun des dessins montre une tortue différente (disque noir) qui peut avancer en suivant les motifs symbolisés par des lignes colorées.

ré-initialisation) du dessin. Autrement dit, il faut garder en mémoire la suite des instructions qui ont été effectuées par la tortue depuis la dernière initialisation.

- INIT : remet la tortue à son point de départ et oublie tout ce qui a été fait depuis la dernière initialisation.

Avec ce principe, vous aurez noté qu'une même série d'instructions aura un effet différent selon l'ensemble de motifs associé à une tortue (cf Figure 3).

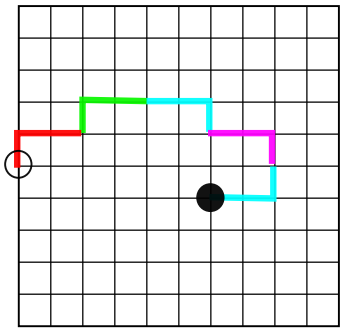
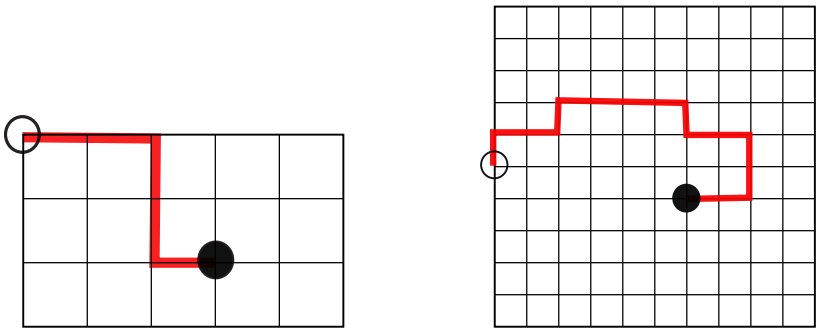


FIGURE 2 – Un chemin coloré dessiné par une tortue utilisant l'ensemble de motifs à droite dans la figure 1

Par défaut, l'application propose au moins un ensemble de motifs ordonnés mais il doit être possible d'en ajouter d'autres.

Il doit être possible de sauvegarder le dessin effectué par la tortue dans un fichier (soit sous forme de dessin vectoriel [3], soit sous forme d'un fichier texte contenant l'historique des commandes) et de recharger un dessin déjà partiellement réalisé. L'avantage du deuxième format est de conserver non seulement le dessin mais aussi l'historique.

3. Il existe probablement des bibliothèques java pour cela.



(a) Tortue associée à l'ensemble de motifs représenté sur la figure 1a

(b) Tortue associée à l'ensemble de motifs représenté sur la figure 1c

FIGURE 3 – Les chemins rouges sur les deux dessins correspondent à la suite d'instructions réalisés à partir de deux ensembles de motifs différents : GO GO TURN GO GO TURN TURN TURN GO. Le cercle noir représente le point de départ de la tortue, le disque noir son point d'arrivée. Chacune des tortues a démarré avec le deuxième motif de la liste.

2 IHM

Pour réaliser des dessins avec la tortue, l'application offre plusieurs fenêtres (autrement dit plusieurs vues) avec des systèmes de saisie différents (pour différents types d'utilisateurs).

Chacune de ces fenêtres est découpée en plusieurs zones :

- la zone « dessin », consacrée à la visualisation du dessin,
- la zone « caractéristiques » qui affiche les propriétés de la tortue :
  - représentation de l'ensemble des motifs associés à la tortue,
  - couleur courante,
  - motif courant,
- la zone « commandes », donnant accès à des fonctions de base :
  - (ré)-initialisation (le dessin est effacé, la tortue remise à sa position initiale, la zone contenant les instructions effacée),
  - annulation de la dernière instruction,
  - rejeu du dessin depuis le début : concrètement, lorsque l'utilisateur clique sur ce bouton, le dessin est réinitialisé, la tortue se positionne à son point de départ et l'ensemble des instructions est exécuté de bout en bout, automatiquement, avec un laps de temps entre le dessin de chaque motif,
  - sortie de l'application.
- la zone « instructions », dédiée à la saisie et à la visualisation des instructions. Dans tous les cas, on peut en permanence voir l'ensemble des instructions exécutées depuis le lancement de l'application ou depuis la dernière ré-initialisation.

Les vues diffèrent sur cette quatrième zone.

En mode « débutant » (cf Figure 4), l'utilisateur construit le programme de dessin pas à pas, autrement dit, il exécute une instruction (et vérifie son effet) avant d'ajouter la suivante.

La zone d'instructions est découpée en deux parties. Une zone de texte **non éditable** permet de visualiser les instructions, tandis qu'un deuxième bloc est dédié à l'ajout de nouvelles instructions au programme de la tortue.

Il existe 3 vues pour cette zone, illustrée par les figures 6, 7, 8, dans l'ordre croissant de difficulté d'utilisation.

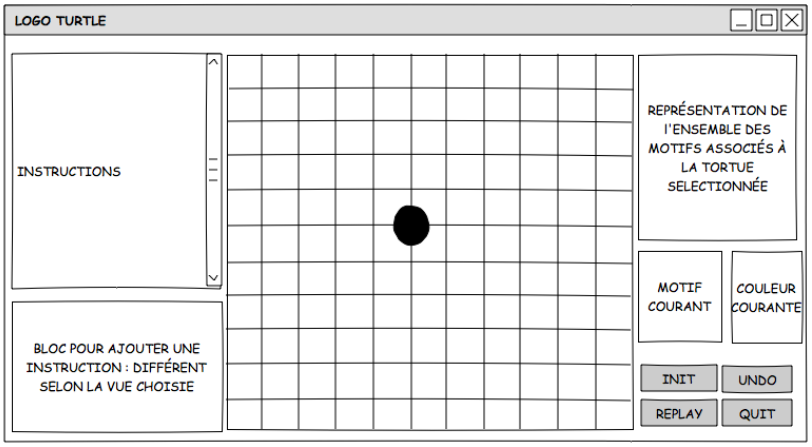


FIGURE 4 – Maquette de la fenêtre de l'application en mode « débutant »

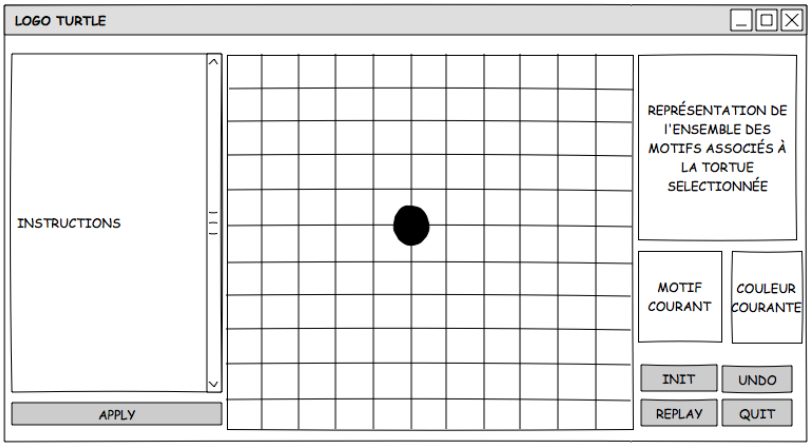


FIGURE 5 – Maquette de la fenêtre de l'application en mode « expert »

Dans la première vue (Figure 6), la plus simple du point de vue de l'utilisateur, chaque instruction est représentée par un contrôle différent. Chacune des instructions GO et TURN est associée à un bouton. Chaque bouton est soit relié à un champ de texte (Figure de gauche) soit à un slider avec des valeurs pertinentes (Figure de droite) pour d'éventuels arguments. Un bouton radio pour DRAW. Des labels colorés pour choisir parmi un nombre limité de couleurs.

FIGURE 6 – Vue 1 du bloc de saisie des instructions (deux possibilités) en mode débutant

FIGURE 7 – Vue 2 du bloc de saisie de instructions en mode débutant

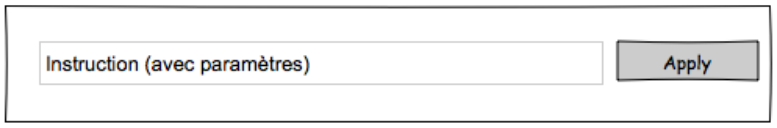


FIGURE 8 – Vue 3 du bloc de saisie de instructions en mode débutant

Dans la deuxième vue (Figure 7), une combo box permet de choisir une des 4 instructions [4](GO, TURN, DRAW, COLOR), et la zone de texte d'indiquer éventuellement le paramètre correspondant.

La troisième vue, enfin, (Figure 8) ne fournit qu'un champ de texte dans lequel l'utilisateur doit écrire à la main l'instruction et ses paramètres.

À noter que lorsque la vue 1 est utilisée, on sait si on est en train de dessiner ou pas. Ce n'est pas le cas sur les vues 2 et 3, et, par souci d'ergonomie, l'application doit prévoir quelque chose pour signifier à l'utilisateur s'il est en mode dessin ou non (par le biais d'une décoration sur la fenêtre ou sur la tortue par exemple).

Évidemment, quelle que soit la vue choisie, dès lors qu'une instruction est correcte et validée, elle est exécutée par la tortue sur le dessin. Elle apparaît également à la fin de la zone de texte contenant l'ensemble des instructions et les informations de couleur et de mouvement courant sont éventuellement mises à jour.

En mode débutant, il n'est pas question que l'utilisateur puisse écrire directement dans la zone de texte contenant l'ensemble des instructions. Il est obligatoire de passer par la zone de saisie spécifique pour ajouter une instruction. Cependant, pour la vue 3, l'instruction entrée apparaîtra simultanément et automatiquement à la fin du bloc des instructions (et si elle est modifiée, les modifications devront également apparaître simultanément dans la zone de texte). Mais seule la validation par le bouton Apply lancera le dessin effectif.

Dans le mode « expert » (cf Figure 5), il n'est pas obligatoire de tester chaque instruction : il est ainsi possible d'en écrire plusieurs (une par ligne) avant de les exécuter. L'organisation de la fenêtre

4. et évite donc en partie les fautes de frappe

est donc un peu différente de la précédente puisque la zone de saisie et la zone de visualisation des instructions sont confondues. Cette fois, l'utilisateur écrit directement les instructions dans la zone de texte, une par ligne. Lorsqu'il veut les exécuter, il suffit d'actionner le bouton « APPLY ». A noter que, comme précédemment, le bouton « UNDO », annule juste la dernière instruction (pas le dernier ensemble d'instructions saisies).

Lorsque l'utilisateur lance l'application, il peut choisir la vue qu'il souhaite utiliser.

Toutes les fenêtres disposent d'un menu qui permet de modifier la taille de la grille de dessin (nombre de colonnes et/ou nombre de lignes), de modifier la configuration de départ d'une tortue (position, mouvement élémentaire, couleur de trait), de sauvegarder le « dessin » réalisé dans un ou plusieurs format(s), de charger un dessin précédemment commencé, de charger un nouvel ensemble de motifs pour la tortue...

3 Travail demandé

Ce projet doit être réalisé en binôme. Pour l'interface, vous ne devez utiliser que les classes standard de Swing (pas d'autres composants tout faits). Le code doit être propre, modulaire, bien indenté, documenté. Les consignes données en début de semestre s'appliquent...

Pour ce projet, vous devez :

- « Figures imposées »
  - réaliser le noyau fonctionnel de l'application (avec un ensemble de motifs codé en dur [5]),
  - implémenter deux vues (au choix parmi celles proposées) : pour ce faire, vous devez respecter l'agencement proposé pour les fenêtres mais les couleurs, les décorations de boutons et autres aspects esthétiques sont laissés à votre libre choix.
  - gérer les actions de la tortue
  - gérer les actions sur les boutons INIT, UNDO, QUIT
  - proposer une fenêtre d'accueil au programme qui permet de choisir la vue qui sera affichée et la taille de la grille.
- « Figure libre »
  - choisir une autre fonctionnalité parmi celles de l'application idéale et l'implémenter (rejeu du dessin, sauvegarde / chargement d'un dessin, une autre vue de l'application, ajout d'un nouvel ensemble de motifs, ajout d'un menu...).

Attention, vos fenêtres doivent pouvoir être redimensionnées jusqu'à un certain point sans que cela ne provoque de « catastrophes ».

Votre code doit être aussi « propre » que possible : pas de redondance de code, un code lisible, modulaire, commenté... Oui je l'ai déjà dit.

Votre application doit être ergonomique et doit autant que possible respecter les critères de Bastien & Scapin qui vous sont fournis sur UPdago.

L'ensemble des sources appartiendra au package `turtle`, plus précisément, le noyau fonctionnel dans le package `turtle.model` et l'IHM dans le package `turtle.ihm`. Libre à vous d'ajouter autant de sous-packages que nécessaires.

Vous rendrez un rapport d'au plus 20 pages, au format pdf (à l'exclusion de tout autre), expliquant comment est structurée votre application, comment vous avez résolu les problèmes de conception rencontrés et comment votre application peut être étendue pour produire l'application rêvée.

Votre travail sera rendu sur UPdago dans une archive `nom1_nom2.tar.bz2`, où `nom1` et `nom2` sont (évidemment) vos noms. À l'intérieur de cette archive, on trouvera le répertoire `nom1_nom2` qui contien-

5. Pour faire les choses un peu proprement, vous pouvez créer une classe « factory » qui fournit une méthode statique renvoyant l'ensemble de motifs de votre choix (même principe que BorderFactory par exemple).

dra le fameux rapport et le package `turtle` contenant votre code source.

Nous nous réservons le droit de convoquer certains binômes à un oral pour d'éventuelles précisions sur le travail rendu.

Il est bien entendu qu'on attend un travail original (c'est à dire personnel) et équitablement réparti au sein de chaque binôme.

À noter enfin que les différentes consignes données dans les documents déjà distribués s'appliquent à ce projet (bis repetita).

4 Évaluation

Les critères pris en compte pour la notation [6] :

- les différentes consignes sont respectées (structuration et format de l'archive rendue, encodage des caractères...) ;
- le code est propre, modulaire, lisible, documenté... ;
- le code rendu compile et le programme peut être lancé et manipulé ;
- l'application est ergonomique ;
- si vous avez repéré des erreurs dans votre application, sans avoir le temps de les corriger, elles sont documentées dans le rapport [7] ;
- le rapport donne les informations nécessaires pour comprendre votre démarche et utiliser votre application ; Il contient notamment la description de votre choix pour la « figure libre »

Les figures imposées (code + rapport) seront notées sur 17, la figure libre (code + rapport) sur 3 points.

6. vous allez dire que je radote
7. Merci de ne pas cacher la poussière sous le tapis.