Tommy De Ganck
Matricule : 000246579
Algorithmes et programmation I
Mnémonique : STIC-B450
Cours dirigé par Anthony Leroy
TP dirigés par Thomas Carette
Année académique 2023-2024
Projet Paint

Rapport sur la création du programme Paint

Permalink du projet

https://gitlab.ulb.be/tdeganck/paintproject/-/blob/1ffd8fe7de715ac50a6cc7c679963ef996ab472c/paintbaseligne.pde

Explications

L'objectif du travail est la réalisation d'un programme similaire à Paint. J'ai commencé le travail par une réflexion sur papier afin d'évaluer les différents éléments dont j'aurais besoin et prévoir les grandes étapes de mon travail.

J'ai commencé par les éléments les plus simples, à savoir les éléments statiques comme la configuration de la fenêtre et du cadre de dessin et de l'outil ligne. J'ai choisi de créer ces éléments avec un rectangle définit par les coordonnées de son centre. Cela permettait un usage plus facile d'une marge unique afin de définir sa taille, et dont la valeur pouvait facilement être réutilisée plus tard pour définir la taille des boutons par exemple. J'ai choisi la taille des boutons en conséquence et j'en ai créé un premier. J'ai porté une attention à définir des variables afin de pouvoir changer très facilement les paramètres principaux du programme en une seule fois. J'ai ensuite configuré l'usage de l'outil ligne, à ce stade comme outil par défaut.

La seconde étape a été de réfléchir à et mettre en place une vrai fonctionnalité de bouton d'outil pour créer des formes. J'ai à nouveau commencé par l'outil ligne, afin de tout d'abord me concentrer sur la façon de gérer l'activation des bouton et les contraintes à imposer pour ne dessiner que dans le cadre de dessin. J'ai choisi de gérer ces événements en mettant en place des « interrupteurs » avec des variables booléennes. Ces interrupteurs m'ont paru pratiques puisqu'ils sont combinables et permettent d'ensuite facilement effectuer des opérations de contrôle afin de déterminer si une forme doit être dessinée et si oui, laquelle.

Une troisième étape, plus compliquée, a été de trouver comment dessiner un rectangle dont les coordonnées de départ sont les positions de la souris au moment du premier clic dans le cadre de dessin. J'ai passé beaucoup (trop) de temps sur cette question. J'ai tout d'abord envisagé et testé l'idée d'enregistrer les valeurs de « mouseX » et « mouseY » dans des tableaux, mais cela ne fonctionnait pas, car nous ne connaissons pas à l'avance la quantité de valeurs à stocker, ce qui fait que nous devrions faire une boucle potentiellement infinie ce qui produit un blocage. J'ai aussi envisagé l'idée de prévoir un espace correspondant à la taille du cadre de dessin, mais cela me paraissais une solution bancale et couteuse. Nous n'avons en effet besoin que de stocker 2 valeurs. J'ai alors regardé plus précisément les fonctions lié aux événements de souris. J'ai constaté que le « mousePressed » n'est appelé qu'une seule fois et que donc on peut y enregistrer les valeurs de la position de la souris au moment du premier clic sans varier ensuite.

J'ai dans une quatrième étape testé cette solution de façon isolée (en dehors de l'ensemble du code, dans un autre sketch). La solution a fonctionné dans ce contexte réduit, mais n'a rien donné quand je l'ai intégrée à l'ensemble de mon sketch « paintbaseligne ». Après avoir vérifié plusieurs fois mon code et testé différentes configurations du code, j'ai remarqué qu'il y avait un souci avec le dessin des boutons lorsque les formes devaient être dessinées. J'ai décidé de créer une fonction chargée de la création des boutons afin de clarifier le code et éviter l'entremêlement. Jusqu'ici, je n'avais pas créé de fonctions, le code était donc long et difficile à lire. La création de la fonction a résolu le problème.

Après cela, j'ai dans une cinquième étape configuré l'affichage des noms des boutons. Pour ce faire, j'ai travaillé dans la fonction d'affichage des boutons. J'ai créé une variable pour indexer les numéros des boutons. L'index était nécessaire pour déterminer le texte à afficher dans chaque bouton créé au sein de boucles imbriquées.

Dans une sixième étape, j'ai optimisé le bouton d'effacement du dessin. Ce bouton était initialement constitué comme les autres boutons et était activé par un premier clic et désactivé par un second. Cette configuration n'était pas pratique car l'effacement se fait d'un coup et le fait de devoir cliquer une seconde fois est particulièrement pénible et peu ergonomique. J'ai alors décidé de créer une désactivation automatique du bouton après effacement en utilisant la fonction « mouseReleased ».

Une septième et dernière étape a consisté en la relecture du code et des commentaires.

Conclusion

J'ai pu implémenter les fonctionnalités minimales demandées pour le projet. Je n'ai toutefois pas assez fait usage des fonctions. Je n'ai ainsi pas entièrement tiré parti des facilités prévues par la création et le choix des variables de départ et avec lesquelles j'ai constitué les boutons et le cadre de dessin. L'usage des fonctions est aussi indispensable pour permettre une fonctionnalité d'effacement de la ou des dernières formes. Si celles-ci sont constituées en fonction, il est possible de stocker leur valeur en bloc par forme dans des vecteurs et donc de les effacer ou rappeler.

Les fonctions à créer serait notamment : « « (dés)activer les boutons », « indiquer le bouton actif », « dessiner une forme ». Je manque de temps pour mettre en place des fonctions correctement avant la remise du 22 décembre, mais je vais continuer le travail pendant les congés afin d'avancer.