Patrice GALLANT et Roberto NIGHTINGALE CASTILLO

PROJET SYNTHÈSE

420-C61-VM

Groupe 00001

Rapport de mi-mandat:

C’est fou. Riez!

Travail présenté à

Jean-Christophe Demers

Cégep du Vieux-Montréal

Départements d’informatique

27 avril 2023

Notre projet s’intitule C’est fou. Riez! C’est un jeu de mot sur le nom du mathématicien qui a élaboré l’algorithme au cœur de notre projet, Joseph Fourier. Voici où nous en sommes rendus :

Nom : Patrice Gallant

* Description fonctionnalité réalisée (même si elle est incomplète) :
  + J’ai réalisé la fonction qui permet de faire tourner les vecteurs dans le modèle, et de les envoyer à la vue pour qu’elle les imprime. Sous la recommandation de notre professeur, j’ai inséré les informations des vecteurs dans une matrice de 5 colonnes, 2 pour les coordonnées polaires de nos vecteurs, une pour l’angle modifié, et les deux finales pour les coordonnées cartésiennes de nos vecteurs avec leur angle modifié. Les modifications se font à l’intérieur de deux boucles for imbriquées qui servent à modifier les vecteurs selon un ordre d’alternance positif-négatif. L’angle est modifié par un nombre de radians déterminé par un système comptant le temps passé depuis le dernier appel de cette fonction. Après cette double boucle, puisque la vue veut imprimer les vecteurs un après les autres, les coordonnées cartésiennes de chaque vecteur sont additionnées avec celles du précédent, afin que le dessin du vecteur commence là où le précédent fini. On envoie alors les deux dernières colonnes de la matrice à la vue.
* Plus grand défi de cette réalisation :
  + Le plus grand défi de cette fonctionnalité a été au niveau de la compréhension. Il fallait comprendre comment chaque partie du problème interagit avec les autres afin de bien pouvoir les assembler. Il fallait comprendre comment le pseudo-chronomètre affecte le mouvement dans l’intervalle, parce qu’elle affecte le changement en radians de l’angle des vecteurs. Le coefficient de nos vecteurs devait être pris en compte, puisqu’il affecte la vitesse à laquelle ils tournent. Il fallait comprendre comment l’interface de dessin de Qt fonctionne afin de savoir qu’il nous fallait additionner les coordonnées cartésiennes des vecteurs l’un à la suite de l’autre. Une fois toutes les parties de la fonction comprises, il a suffi de les assembler.
* Où se trouve dans le code cette fonctionnalité (fichier et numéro de ligne) :
  + Le code de la fonctionnalité se trouve dans le dossier Model, dans le fichier vector\_manager.py, aux lignes 37 à 57.
* Décrivez l’un des développements à venir que vous jugez importants et que vous prendrez en main :
  + Un des widgets que nous voulons absolument intégrer à notre travail est la galerie de dessins. Pour pouvoir l’intégrer, il va falloir donner à l’utilisateur l’option de sauvegarder les dessins, et coder l’accès à la base de données du modèle, pour qu’il puisse y mettre les dessins et les reprendre. Les dessins de la galerie vont pouvoir être réordonnés avec un drag-and-drop. Pour ce faire, ils vont être placés dans une liste chainée.
* Pour ce développement, quel sera le défi le plus significatif :
  + Le plus grand défi de ce développement sera soit de faire fonctionner la liste chainée avec l’interface pour que le changement d’ordre des dessins se fasse de façon élégante, ou la création de l’effet drag-and-drop. Il est difficile de dire combien l’effet drag-and-drop sera dur à faire car c’est un effet que l’on n'a pas fait dans le contexte de Qt auparavant.

Nom : Roberto Nightingale Castillo

Description fonctionnalité réalisée (même si elle est incomplète) :

* La fonctionnalité que j’ai réalisée est : la classe « DrawingAnalyzer » qui effectue la décomposition d’un dessin qui est comme un long trait constitué de plusieurs segments. Plus précisément, ma réalisation analyse tous les points du dessin, à l’aide d’une boucle pour bâtir une matrice « Numpy » munie de 5 informations. Pour chaque point on y trouve : ses coordonnées « x » et « y », la longueur entre celui-ci et le précédent, la longueur entre celui-ci et le premier point du dessin et finalement le pourcentage du dessin rendu au point. Les informations de cette matrice servent à déterminer les coordonnées des points intermédiaires voulus ou en d’autres mots la précision du dessin. Pour cela, j’ai fait la méthode « get\_intermediary\_points ». Ayant la longueur totale du dessin, celle-ci s’occupe de la diviser en "n" parties égales. Ensuite, il faut trouver les cordonnées de ces parties qui sont entre les segments du dessin. Cette étape est l’interpolation des points. La méthode « interpolate » retourne les coordonnées des points intermédiaires du dessin. Grâce à une boucle, une matrice, étant une instance nommée « self.\_\_intermediary\_points », est remplie avec les coordonnées que la méthode retourne. Maintenant que nous avons les coordonnées, nous pouvons calculer le sens, la vitesse, l’angle de départ et la longueur des vecteurs qui feront l’animation avec l’algorithme de transformée rapide de Fourier.

Plus grand défi de cette réalisation :

* Cette réalisation m’a demandé un niveau important de logique et compréhension pour réussir à décortiquer un dessin constitué de plusieurs points. Même si j’avais déjà une bonne idée pour arriver à mon but, il y a des contraintes rencontrées plus tard dans le chemin qui m’ont ralenti. Fragmenter le dessin continu en plusieurs petits segments, à l’aide des points, était plutôt logique. Trouver les coordonnées des points intermédiaires, à l’intérieur de ces segments, a demandé plus de réflexion. Plus précisément, lorsque je fragmente mon dessin en parties égales selon la précision, j’obtiens les distances où devraient se trouver mes points d’interpolation. Ensuite, il faut que je détermine entre quels couples de points du dessin se trouve les points d’interpolation recherchés. Grâce au support que j’ai eu en classe, j’ai pu me servir d’une formule pour trouver à quelle longueur se trouve mon point intermédiaire dans un segment ainsi que ses coordonnées en rapport au dessin.

Où se trouve dans le code cette fonctionnalité (fichier et numéro de ligne) :

* Le code de la fonctionnalité en question est situé dans le dossier Model, dans le fichier model.py à la ligne 58 et termine aux alentours de la ligne 130.

Décrivez l’un des développements à venir que vous jugez importants et que vous prendrez en main :

* Le développement qui va suivre après avoir solidifié ma fonctionnalité sera les contrôles liés à l’animation des vecteurs tournants. Je veux offrir la possibilité de parcourir l’animation pas à pas pour expliquer le comportement des flèches à l’utilisateur, faire pause sur l’animation pour apprécier les flèches enchainées les unes au bout des autres, pouvoir donner le contrôle de la quantité de flèches requises à l’utilisateur en temps réel et suivre les petites flèches en gros plan.

Pour ce développement, quel sera le défi le plus significatif :

* Élargir ma compréhension de la librairie « Pyside6 » ainsi que faire beaucoup d’essaie-erreurs pour arriver à mon résultat. Même si, d’un point de vue, ça a l’air simple, ce processus va avoir besoin de quelques heures de pratique pour y arriver. Le plus grand défi sera de lire la documentation et apprendre à partir de celle-ci.

En pourcentage, donnez votre estimation d’avancement total de développement du projet (incluant la phase 0 de conception et planification) :

* Selon ce que nous avions prévu de faire durant notre planification, nous sommes à peu près à 20% du projet. Nous avons complété les algorithmes fondamentaux de notre projet et avons les bases de notre interface. Cependant, nous avions prévu beaucoup de fonctionnalités additionnelles, et nous sommes encore bien loin d’être en mesure d’implémenter certaines d’entre elles. Si nous ne prenons pas compte des fonctionnalités plus optionnelles, que nous risquons de ne pas pouvoir intégrer, nous serions plus proche de 45%.