









Grille d'évaluation

En guise d'autoévaluation, imprimez et remplissez vous-même cette section. Mentionnez une référence (nom de fichier et numéro de ligne) lorsque demandé pour faciliter la vérification par le correcteur.

Spécifications techniques		Oui	Non	Référence (fichier, ligne)
5	Validation du fichier d'entrée (présence des mots-clés, fonction inconnue ignorée)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>Animation.cpp ligne 63</u>
5	Vérification du nombre de paramètres pour les fonctions dans le fichier texte	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>Instruction.cpp</u>
10	Chargement des animations réussi, affichage conforme des animations	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>Tant que le fichier .sf est conforme</u>
5	Possibilité de jouer les animations à l'endroit et à l'envers	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>Animation.cpp ligne 239 et ligne 261</u>
Spécifications de programmation		Oui	Non	
5	Structures de données imposées : <i>Animation et ses sous-structures</i>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>Animation.h List.h</u> <u>forward-list.h</u>
5	Gestion dynamique de la mémoire et libération de l'espace alloué	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u> </u>

Section réservée au correcteur :

/ 10

Organigramme

/ 30

Réponses aux questions

/ 60

Respect des spécifications

/ 100

Total

Q : Diriez-vous que la structure de données imposée dans ce laboratoire est adaptée au problème de la gestion d'une animation graphique? Parlez de modifications que vous apporteriez à la structure de données pour l'améliorer. (15)

A : Oui, car nous ne connaissons pas la taille de l'animation dès le début et devons donc faire de nombreuses allocations de mémoire. Un tableau dynamique aurait pu suffire, mais les nombreux redimensionnements seraient très coûteux, surtout lors de la copie des éléments vers le nouveau tableau. La liste chaînée est donc adaptée car les nouveaux éléments peuvent être joints à la fin de la structure sans avoir à réallouer la structure au complet.

J'ai modifié la structure de manière à la rendre plus modulaire et réutilisable. J'ai séparé les listes chaînées double et simple de leur contenu et en ai fait des templates afin de pouvoir m'en resservir dans le futur. De plus, les frames et les instructions sont des classes distinctes de la classe animation. Ceci m'a permis d'adopter une stratégie « diviser pour mieux régner » pour charger le fichier.

Q : On désire ajouter de nouvelles fonctionnalités au programme de gestion d'animations. On veut avoir la possibilité de rajouter un frame à partir de l'interface graphique (quand l'animation est déjà chargée). De plus, on voudrait avoir accès à une barre de défilement pour arrêter ou repartir l'animation à partir de n'importe quel frame. Est-ce que la structure de données actuelle peut s'adapter à ce genre de spécification? Si oui, dites comment, si non proposez une structure de données alternatives. (10)

A : Oui, les listes chaînées sont justement parfaites pour de l'insertion au milieu de la structure. Ma structure list (liste chaînée double) comprend déjà une méthode pour l'insertion au milieu. Cette méthode se nomme `insert_after`.

Q : Comment serait-il possible d'accélérer le temps de chargement du fichier d'animation par une modification à la structure de données ? (5)

A : Éviter les allocations et les copies inutiles en utilisant des move operators plutôt que des copy operator. `list<T>& operator=(list<T>&& x);` plutôt que `list<T>& operator=(const list<T>& x);`