NOM : _____ MATRICULE : _____

PRÉNOM : _____

Polytechnique Montréal
Département de génie informatique et génie logiciel

INF2990 : Projet de logiciel graphique interactif

Examen technique

Vendredi 22 novembre 2013, 9 h 00 à 10 h 20

Olivier Gendreau
Martin Paradis

Directives :

- L'examen est sur 20 points et comporte 7 questions sur 7 pages, excluant la présente. Vous devez répondre aux questions 1 à 5 et répondre à la question 6 **OU** la question 7.
- La pondération pour la session est de 10 %.
- Aucune documentation ni calculatrice n'est permise.
- Écrivez vos réponses directement sur le questionnaire.
- La durée de l'examen est de 80 minutes.

Question 1 – Patrons de conception (4 points)

a) De quelle façon le patron « Visiteur » permet-il d'améliorer la maintenabilité et l'extensibilité du code ? **(2 points)**
b) L'implémentation du patron « Visiteur » nécessite d'utiliser les déclarations avancées (ou *forward declarations*). À quoi correspondent-elles et quel problème permettent-elles de régler ? **(1 point)**
c) Expliquez comment il est possible d'effectuer des animations à l'aide du patron composite. **(1 point)**

a) En séparant les algorithmes des objets sur lesquels ils sont appliqués, on parvient à réduire l'interface publique d'une classe. Cela permet également de diminuer la complexité d'une classe, de sorte que celle-ci vise à se figer à long terme. Lorsqu'une classe n'est plus modifiée, mais seulement utilisée, elle devient stable. Le patron « Visiteur » permet donc d'isoler le code à maintenir à un ensemble de fonctions n'utilisant seulement que l'interface publique d'une classe stable.

b) Un peu à l'instar des prototypes de fonctions, les déclarations avancées permettent d'indiquer au compilateur qu'une classe déclarée sera définie ultérieurement. Elles permettent de régler le problème d'inclusion circulaire, qui empêche le projet de compiler.

c) Le patron « Composite » permet de conserver un état entre une racine quelconque et ses feuilles. Dans le projet, il est donc possible de positionner un nœud abstrait par rapport à son parent. On peut alors ajouter une transformation supplémentaire à l'objet pour l'animer. Par exemple, pour une voiture, le nœud racine pourrait être la carrosserie et les nœuds enfants seraient les roues. On positionne la carrosserie par rapport au repère du monde et on positionne les roues par rapport à la carrosserie, car elles seront toujours au même endroit par rapport à celle-ci. Par la suite, on peut ajouter une rotation aux roues afin d'obtenir l'animation. Comme le positionnement des roues se fait par rapport à la carrosserie, quand cette dernière sera déplacée, les roues le seront automatiquement.

Question 2 – Physique (4 points)

a) Pourquoi la résolution d'une collision ne peut être effectuée au moment même où elle est détectée ? **(2 points)**
b) Comment pouvez-vous trouver le temps précis d'une collision entre une station spatiale et un astéroïde ? **(1 point)**
c) Lors de la résolution d'une collision, quelle est l'équation de conservation de la quantité de mouvement qui doit être appliquée ? **(1 point)**

a) Parce qu'il faut s'assurer de traiter les collisions séquentiellement dans le temps. Les collisions potentielles seront détectées selon l'ordre des nœuds dans les structures de données, il n'y a donc aucune garantie qu'elles soient détectées temporellement dans le bon ordre.

b) Comme une station spatiale ne bouge pas et que les astéroïdes ont une vitesse constante, on peut déterminer le temps de la collision en calculant le temps pris par l'astéroïde pour parcourir une distance représentant l'enfoncement.

c) $\boldsymbol{p} = m\boldsymbol{v}$ entre le moment avant la collision et le moment suivant immédiatement la collision, soit $m_1\vec{v}_1 + m_2\vec{v}_2 = m_1\vec{v}_1' + m_2\vec{v}_2'$.

Question 3 – Outils de développement (3 points)

a) Nommez 2 des 4 pas de débogages possibles dans Visual Studio et Eclipse. **(1 point)**
b) Considérant que la compilation du code C++ résulte en une librairie dynamique, par quel mécanisme est-il possible de lancer l'application à partir de Visual Studio ? **(1 point)**
c) Les points d'arrêts (ou *breakpoints*) permettent, autant dans Eclipse que dans Visual Studio, de stopper momentanément l'exécution du code à un point précis du programme. Quelles informations importantes sont disponibles au moment de l'arrêt ? **(1 point)**

a)

1) jusqu'au prochain point d'arrêt (*Continue*)
2) jusqu'à la fin de la présente fonction (*Step out*)
3) jusqu'à la prochaine instruction immédiate (*Step into*)
4) jusqu'à la prochaine instruction locale (*Step over*)

b) Il est possible de configurer Visual Studio pour qu'il ouvre l'interface Java par les lignes de commande en mode débogage. Il attache ensuite immédiatement le processus java créé à la librairie dynamique.

c) (0,5 pt par bonne réponse)

- Le nom et le contenu des variables locales au point d'arrêt;
- La pile des appels de fonction (*callstack*);
- D'autres réponses sont acceptées, pourvu qu'elles soient cohérentes.

Question 4 – Édition de la scène (3 points)

a) Quelles sont les étapes nécessaires pour effectuer une rotation de plusieurs objets sélectionnés ? **(2 points)**

b) Pourquoi n'est-il pas recommandé, lors de la suppression, d'effacer les nœuds au fur et à mesure qu'on itère sur l'arbre? **(1 point)**

a)

1) Déterminer le centre de la rotation par la moyenne géométrique du centre des objets sélectionnés;
2) Effectuer la rotation de chacun des objets par rapport à ce centre en suivant les étapes suivantes :
    a) Effectuer une translation de l'objet vers le centre
    b) Effectuer la rotation sur l'objet
    c) Effectuer la translation inverse de celle calculée en a.
3) (Facultatif) S'assurer que tous les objets restent à l'intérieur de la zone de jeu.

b) Parce qu'un retrait (ou un ajout) peut venir invalider les itérateurs utilisés.

Question 5 – Tests unitaires (2 points)

Dans Visual Studio, comment l'exécution des tests unitaires influe-t-elle sur le processus de compilation ? Qu'advient-il de la génération de la librairie dynamique ? **(2 points)**

Elle ajoute une condition supplémentaire pour déterminer si la compilation doit être considérée comme réussie ou non. Si au moins un test échoue, une erreur de compilation est affichée. Celle-ci permet de déterminer rapidement l'endroit où le test a échoué.

Toutefois, il est possible de lancer l'application lorsque la compilation a échoué à cause des tests, puisque la génération de la librairie dynamique a tout de même été effectuée. Sans cela, il n'aurait pas été possible de lancer les tests unitaires.

Question 6 – OpenGL (4 points)

a) Quel système de coordonnées est utilisé lors des cas suivants (il n'est pas nécessaire d'indiquer les unités) ? **(1 point)** :
   1) Les coordonnées d'un clic de la souris
   2) Déplacement de la vue en projection orthogonale

b) Nommez deux raisons pourquoi il est préférable de conserver les transformations d'un objet (translation, rotation, mise à échelle) à l'extérieur des listes OpenGL ? **(2 points)**

c) Pourquoi est-il nécessaire, à chaque affichage d'un objet, de spécifier l'activation ou la désactivation de l'illumination, des textures, etc. ? **(1 point)**

a)
   1) Coordonnées de clôture.
   2) Coordonnées de la fenêtre virtuelle.

b)
   1) Le contenu de la liste est constant, alors toute variation sur le positionnement ou l'orientation d'un objet nécessiterait de recompiler les listes.
   2) Comme les transformations varient d'un objet à l'autre, il faudrait avoir une liste par instance, ce qui vient nullifier le principe des listes OpenGL.

c) OpenGL étant une machine à état, rien ne garantit qu'un état chargé préalablement le soit encore au moment du dessin d'un objet.

Question 7 – Interfaces utilisateurs (4 points)

a) Il vous est demandé de traiter une panoplie d'évènements reliés au clavier et à la souris. Quel patron de conception permet d'en réduire la complexité et comment parvient-il à le faire ? **(1 point)**
b) Le cadriciel initial respectait le modèle MVC (Modèle-Vue-Contrôleur). À quelle partie du cadriciel correspond la composante « Contrôleur » ? **(1 point)**
c) Les gestionnaires de disposition (ou *LayoutManager*) sont utilisés pour permettre d'influencer le positionnement des contrôles à l'intérieur d'un contenant. Nommez et définissez-en deux. **(2 points)**

a) Le patron état (« state ») permet de gérer efficacement le nombre d'outils et le nombre d'évènements différents. Chaque outil devenant un état concret, il est possible de limiter le nombre d'évènements à gérer à un moment précis. Par exemple, lors de l'outil rotation, on ne gère que le mouvement de la souris, il n'est pas nécessaire de gérer le clavier du tout.

b) Le contrôleur est composé du JNI et de FacadeJNI.

c)
1) FlowLayout : les composants sont placés séquentiellement sur une même rangée. Lorsque celle-ci est pleine, on passe à la rangée suivante.
2) BorderLayout : les composants sont ajoutés selon une région (nord, sud, est, ouest, centre).
3) GridLayout : un FlowLayout utilisant une grille dont toutes les cellules possèdent la même taille.
4) CardLayout : il s'agit d'une pile de contenants dont un seul est visible à la fois.
5) BoxLayout : un FlowLayout permettant d'ajouter des composants autant verticalement qu'horizontalement.
6) GridBagLayout : permet d'ajouter des composants de taille variée à des emplacements spécifiques en utilisant un système de contraintes (GridBagConstraints).