Projet C++ - PPGR : Pacman – Partie 2

Remarques préliminaires :

- Ce projet doit être effectué par les mêmes groupes que pour la partie 1. Le ou les fichiers contenant le code source doivent être postés sur l'extranet, par l'un des 2 étudiants de chaque groupe, au plus tard le dimanche 27 novembre 2011 à 23h59.
- Les critères pris en compte pour la correction sont les mêmes que pour la partie 1.
- Chaque groupe remet un travail original : si vous collaborez entre groupes, vous veillerez cependant à garder un caractère original/spécifique à votre programme.
- **Vous devez programmer les volets 1 et 2 dans leur entièreté**. Vous irez ensuite le plus loin possible dans le volet 3.

Objectif de l'application :

Proposer un jeu de Pacman avec une apparence en 3 dimensions. Les règles du jeu sont identiques à celles décrites dans l'énoncé de la partie 1 de ce projet.

Au terme de cette 2ième partie, l'application proposera aussi des vues et des animations en 3D du labyrinthe. L'apparence de ce dernier sera en outre gérée en utilisant un modèle d'éclairage.

Votre soumission devra aussi **impérativement** inclure un manuel de l'utilisateur (un simple fichier txt suffit) expliquant clairement les fonctionnalités des touches du clavier et des boutons graphiques disponibles.

Du point de vue graphique

Volet 1.

1) Redéfinissez les différents éléments graphiques intervenant dans le plateau de jeu pour qu'ils soient en 3D. Vous définirez *au minimum un élément graphique à l'aide de primitives graphiques élémentaires*, les autres pouvant être construits sur base de fonctions de la librairie glut, telles `glutSolidSphere(…)`, `glutSolidCube(…)`, … Les éléments graphiques peuvent être tout à fait différents de ceux en 2D et il n'est plus obligatoire de prévoir un "mouvement de bouche" du Pacman.
2) Stockez les définitions des éléments graphiques définis à l'aide de primitives graphiques élémentaires (y-compris les boutons graphiques) dans des display lists et utilisez celles-ci dans les viewports concernées.
3) Pour les éléments graphiques 3D, veillez à toujours définir les surfaces de manière à ce que la face "FRONT" soit orientée vers l'utilisateur.
4) Etablissez une projection 3D parallèle du labyrinthe dans sa viewport. Etablissez ensuite une vue en perspective du labyrinthe. Enfin, permettez à l'utilisateur de passer d'une projection parallèle à une projection en perspective, et vice versa.
5) Améliorez les déplacements du Pacman et du/des fantôme(s) : ils doivent donner l'impression d'avoir lieu en continu et non plus passer brutalement de la situation avant le mouvement à la situation après le mouvement.
6) Paramétrez le `gluLookAt(…)` et/ou le `glOrtho(…)` de manière à permettre à l'utilisateur de faire se déplacer la caméra en profondeur, dans les 2 directions. L'utilisateur pourra ainsi décider de zoomer sur le labyrinthe ou de s'en éloigner ; cependant on n'autorisera ces mouvements que dans la mesure où l'on ne s'éloigne ni ne s'approche trop du plateau de jeu.

Remarque : La gestion des problèmes de distorsion n'est *pas* demandée en 3D. Cependant, vous veillerez à ce qu'il n'y ait pas de distorsion dans la fenêtre initiale.

Volet 2.

7) Ajoutez les définitions des vecteurs normaux aux primitives graphiques élémentaires intervenant dans la zone de jeu.

8) Etablissez un modèle d'éclairage de la scène 3D, incluant au minimum :

   a) Une source d'éclairage directionnelle avec une composante ambiante et une composante diffuse ;

   b) Un spot positionnel.

   Les effets des deux sources d'éclairage doivent être visibles sur l'écran initial.

9) Permettez à l'utilisateur de faire pivoter le plateau de jeu sur lui-même autour d'un axe vertical passant par le milieu de la "case centrale", en cliquant sur cette case à l'aide de la souris. Un second clic sur la case arrête la rotation.

Volet 3.

10) Introduisez une animation du Pacman lorsqu'il bouge (il ne doit pas forcément avoir de bouche au sens propre du terme).

11) Permettez à l'utilisateur de se mettre à la place du Pacman, soit via le clavier, soir via la souris. On entend par là que la caméra devra se placer à proximité du Pacman à une distance fixe de celui-ci et être orientée dans la direction d'avancement de celui-ci. L'utilisateur devra aussi pouvoir revenir au mode classique.

12) Attention, certains trouvent que vous passez trop de temps sur ce projet, il va être temps de se consacrer un peu plus aux autres cours …

Du point de vue C++

La partie C++ du projet sera évaluée en tenant compte des critères suivants:

- Respecter la programmation modulaire (séparation .h .cpp)
- Gérer la mémoire sans fuite (allocation/libération des ressources)
- Respecter l'encapsulation (usage public/private/protected)
- Respecter le modèle MVC
- Démontrer la maîtrise de l'héritage.

Volet 1

1) La gestion de la 3D se fera en définissant de nouvelles classes View3D et View3DPerspective héritant d'une classe de base View.
2) La gestion du passage d'une vue à l'autre se fera de façon centralisée en changeant simplement le contenu d'une variable globale (ou d'un attribut de classe) de type View *. Le fait que le dessin se fasse en 3D, en perspective ou en 2D sera fonction du type à l'exécution de cette variable via le mécanisme du polymorphisme. Il n'est donc pas question d'avoir un flag is3D et de le tester pour savoir s'il faut appeler une version 3D ou 2D de la vue.
3) Pour savoir quelles actions le contrôleur peut effectuer sur l'objet View (rotation sur son axe par exemple), le type à l'exécution sera testé dans le contrôleur.

Volet 2

4) A chaque complétion de niveau on chargera un labyrinthe différent de celui qui vient d'être terminé.
5) La taille du labyrinthe au niveau 2 doit être plus grande que celle du niveau 1. On veut que les ressources représentant l'ancien labyrinthe soient libérées avant de charger le nouveau labyrinthe.

Volet 3

6) Utilisez les smart pointers de C++11 là où cela est pertinent.