IN204: Projet Tetris

Document technique

Dernière modification: 3/18/2013

Marouane Fazouane & Zhang Mengmeng

Informations générales

- Projet : **Tetris**
- Langage : **C++11/ Qt**
- IDE : **QtCreator**
- Source Control : **GIT** https://github.com/
- Documentation : **Doxygen**
- Modéliseur UML : Visio, Umbrello
- Langue : Français

TODO

- Analyse fonctionnelle OK
- Description des modules OK
- Use Case Diagram OK
- Class diagram OK
- Démonstrateur OK

Roadmap

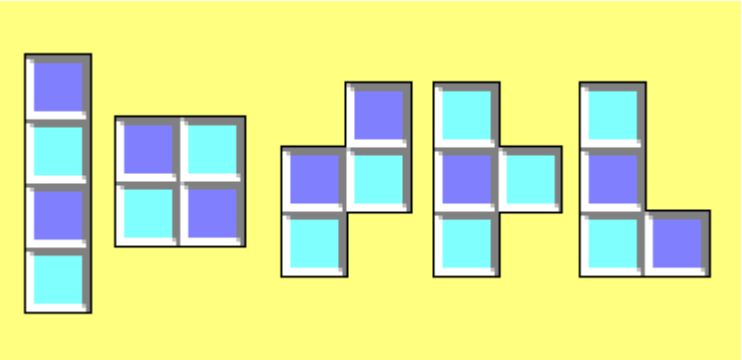
- Installer les bibliothèques sur nos ordinateurs
- Prendre un compte Micro sur github et configurer l'IDE
- Se mettre d'accord sur le CodingStyle
- Compléter la description des modules
- Compléter l'UseCase
- Commencer les diagrammes de classes
- Réaliser un mini Tetris (très simple jeu) en mode console : Premier prototype du jeu
    - But :
        - tester la suppression de lignes ;
        - tester les actions (rotations, déplacements…)
        - tester la chute
        - tester la fin du jeu
    - Description :
        - Carte : matrice
        - Pièces : des positions : x,y (Model) ; des étoiles (View)
        - Commandes données sur la console
- Réaliser un prototype du système de menus
- Approuver les diagrammes et implémenter le diagramme de classe en C++
- Commencer la documentation
- Travailler la GUI basique et l'intégrer au code
- Développer la GUI et le système de menu
- Rajouter la configuration par xml et un gestionnaire de son

Configuration

- Télécharger Qt 5.0.1 for Windows 32-bit (MinGW 4.7, 823 MB) sur http://qt-project.org/downloads
- Ouvrir le fichier Qtetris2.pro (indication, l'icône à cote devrait être verte)
- Rajouter les deux dossiers suivants au Path système :
    - C:\Qt\Qt5.0.1\5.0.1\mingw47_32\bin
    - C:\Qt\Qt5.0.1\Tools\MinGW\bin
- Lancer qmake (menu « compiler » -> « exécuter qmake)
- Menu « compiler » puis « compiler le projet QTetris2 »
- Avant d'exécuter, déplacer les fichiers/ dossiers ci-dessous vers le dossier d'exécution du projet.
    - QTetris.png
    - config.xml
    - assets/

Le Jeu [Tetris]

- Généralités : voir Wikipedia
- Pièces : 5 pièces+ leurs miroirs (les tétrominos)



- Actions possibles :
  - Rotation
  - Déplacement dans le monde (droite-gauche pour de la 2D)
- Evénements possibles :
  - Collision
  - ligne horizontale est complétée sans vide
- Règles du jeu :
  - Une ligne complétée disparait
  - Si l'écran se remplit jusqu'en haut, le joueur est submergé et la partie est finie
  - Faire une seule ligne ne rapporte que 10 points, alors qu'en faire 2 en rapporte 30, 3 lignes rapportent 50 et au-delà de 4 (n lignes) en rapportent 80*n/4
  - Le nombre de points est augmenté à chaque niveau selon l'équation f(p, n)= p(n+1) où p est le nombre de points au niveau 0 et n le niveau.
  - le niveau augmente en fonction du score
  - la vitesse de chute des pièces augmente avec le niveau

Information importante

Dans notre version du jeu, on a voulu créer un effet de réaction en chaine, comme expliqué sur les figures ci-dessous. On a donc opté pour un système de gestion de la scène pour effectuer cette tâche.

Analyse fonctionnelle

1. Fonction principale
   - Un jeu Tetris jouable.
2. Contraintes
   - Nombre de joueurs : 1
   - Menu (start, pause, quitter le jeu, se connecter…)
   - Suivi de score (HighScores…)
   - Type de commandes (rotation, accélération, mouvement gauche/ droite…)
   - Vitesse (qui progresse)
   - Difficulté qui augmente (les niveaux)
   - Type d'entrées : touches de clavier
   - Vérifier la possibilité de faire une action (pour faire une rotation/ se déplacer…)
3. Fonctions complémentaires
   - Système de menu
   - configuration xml
   - Sons
   - UI du jeu extensible (Prévoir et afficher l'élément suivant etc…)

Design Patterns utilisés

- Observer : simulé par le système de signals/slots de Qt,
- Singleton,
- Builder,
- Model-View-Controller (séparation des classes Model des classes Controller, la View et les commandes utilisateurs sont gérées par Qt).

Concepts architecturaux utilisés

- Lazy initialization
- Introspection
- Inversion de dépendance
- Fonctions Lambda

Le code est décomposé en sous-systèmes

- Système de menus,
- un système de gestion de paramètres et de ressources (des objets et des fonctions qu'on « expose ») (XML)
- Gestionnaire de sons
- Le jeu en question (QTetris)
- Un système de suivi de score (SQL)