ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

MetaProjet LOA

Nom :	GENEST Hugo BOREL Alec

Nom du projet CLiiCKeR

1. Objectif

L'objectif de ce projet est vous faire réaliser une petite application utilisant l'API Qt.
Pour ce faire vous allez au préalable définir :

- Les fonctionnalités (ou opérations) que vous souhaitez réaliser dans votre application.
- Les données manipulées par votre application : ce que vous appellerez votre « modèle ».

1. Description des fonctionnalités

- C'est un jeu, où le but est de cliquer sur un bouton principal
- Le bouton principal nous fait gagner de la ressource, le but est d'en gagner le plus possible
- On peut dépenser les ressources pour acheter des objets (amélioration des cliques, auto-cliqueur), et générer encore plus de ressource !
- On peut également acheter des améliorations pour ces objets, et les rendre encore plus efficaces !
- Les objets et les améliorations coûtent de plus en plus chers, c'est le caractère limitant
- Pour pallier à cette limitation, on peut aheter(créer) une nouvelle fenêtre vierge
- On peut nommer les fenêtres, pour pouvoir les rechercher plus facilement
- On peut sauvegarder l'avancée de la partie dans un fichier, et charger une partie depuis un fichier déjà existant
- _____

1. Description des éléments du modèle

- La fenêtre a un nom dynamique, avec le nom du jeu, suivi de l'argent actuel.
- à gauche, il y a un onglet « Save », se déroulant en trois boutons : « Sauvegarder », « Nouvelle Partie », « Charger une Partie ». C'est de la lecture et écriture de fichier.
- à droite, nous avons la fenêtre actuelle, avec un ENORME bouton, et l'agent actuel indiqué en dessous. Nous trouvons en haut l'ensemble des fenêtres, et une barre de recherche pour pouvoir rechercher les fenêtres par nom. Si il y a trop de fenres, on montre jsute le bouton fermer des autres fenêtres (internet like)
- à gauche, nous avons la liste des objets de la fenêtre actuel, avec le logo, le nom, le prix, et la quantité possédé. On peut dérouler un objet pour accéder à ses améliorations.

2. Travail demandé

Ce projet est à faire en binôme en utilisant l'API QT.

Implémentez un programme de gestion de score de cliques.
Vous veillerez donc à séparer clairement le modèle ( ) des vues ( ) et des contrôleurs (les widgets et/ou les délégués permettant de modifier le(s) modèle(s)). Chaque classe devra être documentée (avec Doxygen) et son auteur identifié.

1) Concevez les classes permettant de représenter votre modèle. Les classes directement en relation avec l'interface graphique pourront être implémentée sous la forme de QObjects afin de fournir des slots à l'interface et des signaux vers l'interface.
2) Concevez l'interface graphique en QT permettant de gérer les éléments de votre modèle.
3) Concevez les classes nécessaires à l'importation et l'exportation de votre modèle dans divers formats :

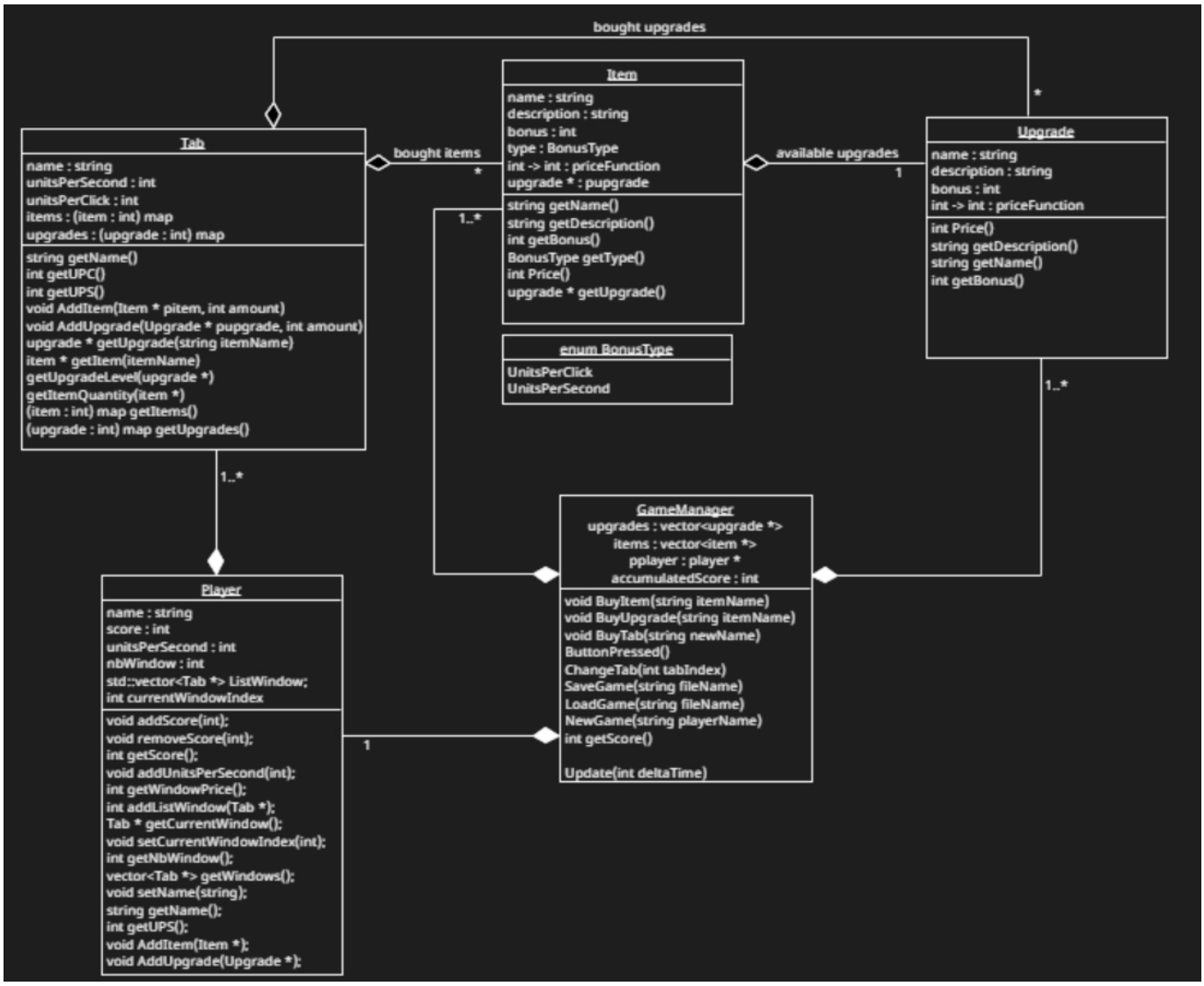
a) Votre propre format.
b) Le format .

1. Conseils

- Les classes directement en relation avec l'interface graphique pourront être implémentées sous la forme de QObjects afin de fournir des slots à l'interface et des signaux vers l'interface. La classe contenant les données devra respecter l'interface du modèle choisi.
- Evitez d'ouvrir une boite de dialogue à chaque fois que vous souhaitez modifier un champ, préférez l'édition directe du champ (grâce à l'éditeur d'un délégué par exemple).
- Lors de l'édition d'un champ, utilisez un QValidator (ou un de ses descendants) pour vérifier la validité de ce que vous tapez.

- Si vous développez vos propres widgets (pour la visualisation et l'édition d'une donnée), n'oubliez pas de lui associer des propriétés (qui apparaîtront alors dans le designer) et des signaux/slots pour les connecter au(x) modèle(s).
- Vous pouvez si vous le souhaitez utiliser la petite application qui vous est fournie (dans /pub/FISE_LAOA24/projet/CookingClock-<date>.zip) pour vous aider à :
    - Rendre votre application réellement multiplateforme.
    - Mettre en place une icône pour l'application (sur toute plateforme).
    - Charger et sauvegarder les préférences de l'application.
    - Traduire votre application dans différentes langues et changer la langue de votre application à la volée.
    - Mettre en place une boite de dialogue « à propos … » permettant de présenter l'application, ses auteurs et les crédits nécessaires si vous avez « emprunté » des ressources comme des icônes ou des librairies par exemple.
    - Avoir un exemple de code documenté avec Doxygen.
    - Avoir un exemple de fichier de configuration (.pro) relativement complet et vous permettant (en plus de compiler votre application) de :
        - Générer une archive transportable de votre code (que vous pourrez me rendre à la fin du projet).
        - Générer la documentation de votre code.
        - Générer et compiler les fichiers de traduction multilingues.

Description détaillée des éléments minimaux du modèle :



Documentation

- Documentation QT : http://doc.qt.io/qt-6/
  - Tutoriel Model / View : http://doc.qt.io/qt-6/model-view-programming.html
    - Une version simplifiée mais en français a été réalisée par Thierry Vaira : http://tvaira.free.fr/bts-sn/qt/cours/qt-model-view.pdf
  - Exemples de lecture/écriture en XML : http://doc.qt.io/qt-6/examples-xml.html
  - Un exemple simple de carnet d'adresses : http://doc.qt.io/qt-6/tutorials-addressbook.html
- Sites contenant des exemples et des tutoriaux QT
  - http://qt-apps.org/ (en anglais)
  - http://qt.developpez.com/ (en français)
- Documentation format standard : _____

Exemple de modèle

Pour que votre modèle soit accepté, il doit être suffisamment complexe : Typiquement, un élément de modèle doit être composé de sous éléments aux arités variables, ces sous éléments pouvant eux aussi être composés de sous-sous éléments. En voici un exemple abstrait :