Université de Strasbourg
Interactions Homme-Machine

Licence d'informatique, semestre 6
Feuille de sujet

Projet d'IHM : gestionnaire de thèmes

Présentation

Vous devez créer une application Qt pour bureau Linux qui permet de gérer des transformations de palettes de couleurs appelés "thèmes". Un thème est défini par un ensemble de pairs de couleurs : une couleur source, et une couleur target (cible). Un thème peut ensuite être appliqué à un ou plusieurs fichiers pour modifier toutes les occurrences de couleurs dont la couleur est source en la couleur target. Cet outil permettrait de modifier facilement toutes les couleurs d'un site web, du code d'une application, ou d'un fichier de couleurs.

Exemple : fichier1 comporte des occurrences de la couleur #0080FF. Le thème A contient la paire de couleur source : #0080FF et target : #AABBFFF0. En appliquant le thème A sur fichier1, les occurrences de #0080FF deviennent #AABBFFF0.

Ce projet est un projet itératif (ce qui signifie que les étapes sont courtes et ordonnées) destiné à vous guider pour apprendre efficacement à travailler une IHM et aussi à apprendre efficacement les bases du C++. Il est conçu pour qu'une séance et un peu de travail hors cours fasse un niveau entier (sauf le niveau 3). Un cahier des charges est donné après le niveau 3.

Vous devrez rendre le rapport de design du niveau 3 avec son *wireframe* sur moodle, et le projet au niveau 9 sur un dépôt gitlab partagé avec l'encadrant. Le rapport de design comptera pour 30% de la note, et le niveau 9 pour 40% de la note.

Installation

Cette section est facultative si vous travaillez sur turing. **Notez qu'il faut que votre application compile et s'exécute sur turing**. Cependant, généralement, un code Qt écrit sur *Windows* fonctionne bien sur *Linux* et vice versa. Le dépôt contient un fichier d'intégration continue qui vérifie automatiquement si le projet compile sur une machine *Linux*.

- Pré-requis : une installation suffisante prend au moins 5 Go de mémoire.
- Suivre le lien : `https://www.qt.io/download-qt-installer`
- Lancez l'application téléchargée. Sur Linux, passez par la console et faites *chmod u+x* sur l'application.
- Faites le même exercice avec l'application téléchargée.
- Suivez l'installation. Vous avez besoin d'un compte.
- Dépliez la dernière version stable : pour une installation confortable, cochez *Desktop gcc 64-bit* et *Qt Debug Information Files* (facultatif ; permet de visualiser le code de Qt pendant le déboguage, sans le code assembleur, mais prend beaucoup d'espace).
- Dépliez les outils de design et de développement : QtCreator va s'installer. CMake peut être utile. Assurez-vous que Qt Design Studio est coché.
- Si pendant le développement vous n'avez pas tout ce qu'il vous faut, vous pouvez relancer l'application téléchargée pour rajouter des modules.

Niveau 1 Prise en main et programmation C++ (2 points)

Objectifs d'apprentissage

- Employer QtCreator pour compiler et modifier un projet simple en C++/Qt
- Découvrir les mécanismes de base du C++ (classes, méthodes) grâce à une mise en pratique

Commencez par récupérer, ouvrir et compiler le projet :

- `https://git.unistra.fr/nicolas.lutz/color-theme-manager` : Faites un *fork* et mettez votre encadrant en *Reporter*.
- Clonez votre *fork* sur votre machine.
- Ouvrez le fichier `app.pro` avec QtCreator.
- Configurez le projet : choisissez notamment un dossier pour la compilation en mode Debug et en more Release, préférablement en dehors du dossier de source. Vous pouvez revenir sur les étapes de compilations plus tard. Les données de compilation ne concernent qu'une machine et sont dans le fichier `.pro.user`.
- Assurez-vous que tout compile et s'exécute correctement. Si une fenêtre à peu près vide s'ouvre, c'est que c'est bon.

En interactions homme-machine, une façon de programmer efficacement est de séparer le modèle, la vue et le contrôle dans des parties distinctes du code (voir `https://doc.qt.io/qt-5/model-view-programming.html`). Vous allez coder une partie du modèle et vous familiariser avec le C++ sur le tas.

Créez une classe `ColorPair`, représentant une pair de couleurs.

- Mettez les fichiers .cpp et .h dans les dossiers `src/` et `include/` respectivement.
- La classe prend comme attribut privé un identifiant de type `QString` appelé `m_id` et deux `QColor` appelés `m_color1` et `m_color2`.
- Implémentez les constructeurs publics suivants :

```
ColorPair(); //constructeur vide
ColorPair(const ColorPair &color); //con. par copie
ColorPair(const QString &id, const QColor &color1, const QColor &color2);
ColorPair(const QString &id); //con. initialisant uniquement l'identifiant
```

Créez une fonction publique statique qui transforme une `QColor` en `QString` au format hexadécimal "#rrggbbaa" (a est l'alpha) grâce à la fonction `name` de `QColor`. Créez également une fonction qui fait l'inverse : transforme un code `QString` "#rrggbbaa" en `QColor` et renvoie celle-ci.

```
static QString toRGBA(const QColor &color);
static QColor fromRGBA(const QString &colorStr);
```

Quelques raccourcis pour QtCreator :

- `F2` : basculer entre définition/déclaration d'un symbole
- `F4` : basculer entre source/en-tête d'un fichier
- `Ctrl + Shift + R` : Modifier le nom d'un symbole (se répercute sur toutes ses occurrences)

Niveau 2 Un peu plus de C++ (2 points)

Objectifs d'apprentissage

- Utiliser des opérateurs, des foncteurs et des conteneurs en C++
- Concevoir un lecteur de fichier XML avec Qt

Nous allons créer une classe qui contient un ensemble de pairs de couleurs, pour lire un fichier HTML et en extraire les couleurs.

En C++, il est possible de surcharger certains opérateurs : par exemple, l'opérateur "+" de QString crée un nouveau QString en rajoutant le contenu d'un autre QString à la fin du premier. Pour savoir quel prototype un opérateur doit prendre, référez-vous à cette page : `https://en.wikipedia.org/wiki/Operators_in_C_and_C`.

En C++, on appelle "foncteur" un objet surchargeant l'opérateur "()". Ils peuvent être pris en paramètres pour appliquer automatiquement une fonction sur d'autres objets.

- Dans les fichiers `colorPair.h` et `colorPair.cpp`, créez un foncteur `CompareColorPair` servant à comparer deux `ColorPair`. L'opérateur "()" doit renvoyer true si le nom de la première pair de couleurs passe avant celui de la seconde dans l'ordre lexicographique. Utilisez une fonction ou un opérateur de QString.
- Créez une classe `XMLReader` prenant en attributs privé un `set` de `ColorPair` utilisant le foncteur de comparaison `CompareColorPair` (voir `https://en.cppreference.com/w/cpp/container/set`).
- faites en sorte de pouvoir lire les fichiers XML donnés sur moodle grâce à une méthode `read(const QFile&)`.
- Testez vos fonctions dans le `main`, ou de façon pérenne en créant un nouveau fichier exécutable de test contenant son propre `main`.
- Si les besoins d'implémentations l'exigent, modifiez les niveaux 1 et 2 selon vos envies, en changeant par exemple de conteneur.

Quelques autres indications en vrac pour vous aider en C++ :

- Il existe une excellente charte pour savoir quel conteneur utiliser pour chaque cas d'utilisation : `https://i.stack.imgur.com/G70oT.png`
- Le caractère & sert aussi à désigner une référence : `https://en.cppreference.com/w/cpp/language/reference`
- Faire une allocation dynamique avec le mot-clé `new` : `https://en.cppreference.com/w/cpp/language/new`
- Créer un constructeur avec une liste d'initialisation des membres : `https://en.cppreference.com/w/cpp/language/constructor`
- Les fonctions lambda en C++11 servent à écrire compactement un ptr. de fonction : `https://en.cppreference.com/w/cpp/language/lambda`
- Écrire `const` derrière le prototype d'une méthode indique au compilateur que la fonction ne modifie pas (et ne doit pas modifier) les variables membres de l'instance de la classe.
- La définition d'une méthode `f` d'une classe `A` s'écrit comme une fonction en `C`, mais le nom devient `A::f` pour signifier l'appartenance à la classe `A`.

Niveau 3 Wireframe et rapport de design (un rendu sur 30%)

Objectifs d'apprentissage

- Créer le squelette d'une interface graphique grâce à un outil de dessin d'interface
- Créer un rapport de design accompagnant un *wireframe*.

Ce niveau, en **deux séances**, consiste à concevoir le *wireframe* de l'interface utilisateur (*UI*) de votre application à l'aide de `app.diagrams.net` (aussi connue sous le nom de `draw.io`).

Le *wireframe* est une vue du squelette de l'interface graphique, sans images (voir *wikipedia* : *website wireframe*). Elle ne contient pas de style typographique, de couleur, de graphismes, mais exhibe les fonctionnalités (uniquement de l'*UI*, pas du projet entier), le comportement (que se passe-t-il quand on effectue une action), et la priorité du contenu (par la position des éléments, la taille de la police...). Vous trouverez de nombreux exemple en ligne, par exemple celui-ci : `https://bit.ly/3rzTkvt`. Il s'accompagne du rapport de design, en pdf, qui reprend le *wireframe* et le complète avec des commentaires décrivant le flux de travail de votre application.

Le rapport de design ne décrit pas le style, mais permet plutôt de complémenter le *wireframe* pour lever le moindre doute par rapport au rôle de chaque parcelle du *wireframe*, comme par exemple quelle zone regroupe les boutons liées aux actions de la fenêtre principale (s'il y a lieu).

Voici les consignes principales :

- Le *wireframe* est *low fidelity* (*wikipedia*). Les designers ne sont pas tous d'accord pour définir ce qui est acceptable ou non dans un *wireframe* lo-fi, donc voici les consignes : ne mettez pas d'icônes, de style, d'images, de couleurs, de *widgets*. Vous pouvez mettre du *Lorem Ipsum* (mais privilégiez des lignes vierges si possible), des formes, des formes avec des croix, et du texte pour décrire ce que représente une forme. Les barres de défilement, si nécessaires, sont acceptables.
- Pour éviter de perdre du temps à le faire et le refaire, vous pouvez vous entraîner à construire le *wireframe* de *youtube* d'après le site, dont des solutions sont proposées en ligne, avant de faire le votre.
- Utilisez autant de pages que vous voulez pour le *wireframe*.
- À l'inverse, essayez d'être concis pour le rapport de design.
- La vue que vous créez est une vue que vous considérez comme étant optimale, mais l'application peut prendre des libertés si le *wireframe* est trop complexe.
- Merci de rendre le rapport de design en *.pdf*.
- **Votre *wireframe* doit répondre au cahier des charges de la page suivante**.

Point sur la notation

Chaque niveau a un nombre de points attribué. Pour avoir tous les points du projet, il faut que le cahier des charges soit entièrement respecté et qu'il n'y ait ni erreur de programmation ni erreur de conception. Il existe de nombreux types d'erreurs de conception possibles. En voici quelques exemples :

- ne pas traiter des erreurs de l'utilisateur
- perdre des données suite à une erreur de l'utilisateur ou de l'application
- concevoir une fonctionnalité inutile
- concevoir une fonctionnalité dont une meilleure alternative est clairement identifiable (notamment mais pas que : moins d'actions utilisateur nécessaires ou moins d'erreurs possibles)
- ne pas informer l'utilisateur d'évènements ou d'erreurs.

Cahier des charges

Les clients sont des utilisateurs maîtrisant la programmation web.

Le support est une application de bureau (pas de multi-support prévu).

Le style n'est pas imposé.

Les fonctionnalités sont données ci-dessous et indiquent tout ce que doit pouvoir faire l'application (ce qu'on appelle les actions), ainsi que quelques vues obligatoires.

Actions générales :

- Importer un fichier contenant des couleurs.
- Calculer le thème source d'un tel fichier.
- Charger tous les thèmes importés à une session précédente.
- Importer un thème.
- Créer un thème.
- Sauvegarder tous les thèmes.

La vue principale doit contenir :

- Les thèmes.
- Un endroit pour notifier l'utilisateur avec des messages non bloquants.
- Un endroit pour montrer le progrès d'une tâche de fond.

La vue d'un thème doit contenir :

- Une icône (image)
- Un nom

Actions liées à un seul thème :

- Sauvegarder le thème.
- Mettre à jour un thème d'après un lien internet.
- Modifier ce lien internet.
- Afficher une liste des paires de couleurs source et target.
- Modifier les couleurs.
- Ajouter une nouvelle paire de couleurs.
- Appliquer le thème sur un fichier pour remplacer les occurrences d'une source par sa target.

Pour information :

- L'icône d'un thème devra faire, dans l'application finale, au moins 32x32 pixels.
- Le nombre maximum de thème est indéfini.
- Le cas typique d'utilisation sera d'avoir une dizaine de thèmes.
- Le nombre maximum de couleurs par thème est indéfini.
- Le cas typique d'utilisation sera d'avoir une centaine de couleurs par thème.
- Un utilisateur moyen va typiquement vouloir utiliser l'application pour modifier les couleurs d'un seul fichier ou d'un projet entier.

Niveau 4 Programmation de l'interface et des actions de MainWindow (3 points)

Objectifs d'apprentissage

- Produire une interface dans Qt grâce à QtDesigner
- Programmer le comportement de l'interface grâce au système de `signal` et de `slot` de Qt
- Programmer des interactions entre modèle et interface

Ce niveau consiste à modifier l'interface pour y refléter celle que vous avez conçue, et à modifier le modèle pour que les fonctionnalités soient programmées.

Qt fonctionne avec un système de `signal` et de `slot`. Un *widget* peut envoyer un `signal` pour signifier un changement de statut ou des données à envoyer, et un ou plusieurs autre(s) *widget(s)* peuvent recevoir ce `signal` pour effectuer un traitement appelé `slot`. Un seul *thread* est utilisé par chaque *widget* pour exécuter les `slots`, et ils sont toujours *thread-safe* : pour un même widget, il ne peut y avoir deux `slots` qui sont exécutés en même temps, à condition que l'appel soit effectué par l'émission d'un `signal` (via le mot-clé `emit`). Enfin, l'appel d'un `slot` via l'émission d'un signal ne bloque pas l'application. Plus d'informations : `https://doc.qt.io/qt-5/signalsandslots.html`.

Commencez par modifier la classe `MainWindow`.

- Ouvrez et modifiez *mainwindow.ui* depuis QtCreator pour ouvrir QtDesigner.
- Utilisez QtDesigner pour faire refléter le *wireframe* de la fenêtre principale sur votre application.
  Note : vous pouvez combiner les *layouts* en mettant un *widget* abstrait dans un *layout*, et en attribuant un *layout* à ce *widget* abstrait.
  Note : Qt Designer est parfois capricieux : pour attribuer un *layout* à un *widget* abstrait, il faut que ce dernier contienne un autre *widget* (par exemple un bouton, une liste, un label, etc).
- Créez les actions générales données dans le niveau 3.
- Placez ces actions dans l'interface.
- Dans QtDesigner, clic droit sur chaque action → aller au `slot` → `triggered()`.
- Implémentez les fonctions crées en mettant à jour le modèle. Vous aurez notamment besoin d'une classe *Theme* et éventuellement d'une classe contenant des *Theme*. Vous pouvez stocker les thèmes sauvegardés et à charger dans un dossier situé dans le dossier de compilation.

Niveau 5 Suite de la programmation de l'interface (3 points)

Objectifs d'apprentissage

- Créer sa propre classe d'interface dans Qt
- Créer son propre `signal` et l'utiliser dans Qt

Ce niveau consiste à créer les *widgets* qui apparaissent dans la liste et à finir les fonctionnalités de base.

- Créez une nouvelle classe `ThemeWidget`, avec .cpp, .h, et .ui via QtCreator. La classe va hériter d'un QWidget, ou d'une classe héritant d'un QWidget, selon votre convenance.
- Placez les sources et en-têtes dans le dossier `Widget/` de `src/` et `include/`.
- Créez l'interface graphique du `ThemeWidget` et son code de base, avec notamment un pointeur, une référence, ou un autre moyen d'accéder au thème correspondant dans le modèle. Le moyen que vous utilisez va conditionner très fortement les bugs que vous risquez de rencontrer.
- Faites en sorte qu'en ouvrant l'application, les thèmes sont chargés et visibles dans la liste des thèmes.
- dans `ThemeWidget`, créez un signal `clicked()` et faites en sorte qu'il soit émit par le `ThemeWidget` lorsque l'utilisateur clique sur celui-ci, en ré-implémentant un évènement de la classe.
- Utilisez ce signal comme vous en avez besoin. Un signal peut être récupéré par n'importe quel *QObject*.
- Continuez ensuite le projet en modifiant le modèle et l'interface de façon à ce que l'interface soit entièrement opérationnelle, avec les imports/exports/sauvegardes de thèmes.

Remarque : vous n'avez pas besoin de créer des fichiers .ui si vous préférez passer directement par le code. Certains utilisateurs experts ne s'en servent plus. Un fichier .ui n'est qu'un fichier xml qui est rempli automatiquement par Qt Designer, puis compilé en C++ (et vous pouvez voir le code généré dans le répertoire de compilation).

Niveau 6 Application du thème, *drag & drop* (3 points)

Objectifs d'apprentissage

- Concevoir une fonctionnalité de traitement des données en anticipant le comportement de l'utilisateur.
- Programmer une fonctionnalité de glisser/déposer avec Qt.

Continuez le développement en permettant à l'utilisateur d'appliquer un thème via l'application.

- Faites en sorte de pouvoir modifier toutes les occurences d'une couleur source sous le format "#rrggbb" ou "#rrggbbaa" de fichiers fournis par l'utilisateur, en la couleur target.
- Cette fonctionnalité modifie un fichier existant. Quels sont les risques encourus ? Faites en sorte d'éviter ces risques, ou de minimiser leurs conséquences s'ils sont inévitables.

Mettez en place des fonctionnalités en *drag & drop* (ou glisser-déposer). Voici quelques idées de fonctionnalités potentielles :

- Réorganiser la liste
- Importer un thème
- Ouvrir un fichier à modifier
- Sauvegarder un thème

Niveau 7 Notifications, téléchargement, mise à jour (3 points)

Objectifs d'apprentissage

- Concevoir une fonctionnalité de traitement des données en anticipant le comportement de l'utilisateur.
- Programmer une fonctionnalité de glisser/déposer avec Qt.

Dans un premier temps, faites en sorte de donner autant de notifications utiles que possible à l'utilisateur via la zone de notifications. Faites en sorte de mettre le progrès à jour lors des tâches qui prennent une durée indéterminée, ici et pour tous les prochains niveaux.

- Faites en sorte de pouvoir télécharger un thème et de pouvoir le mettre à jour. Vous pouvez vous "inspirer très fortement" des exemples fournis par Qt.
- Faites en sorte que le téléchargement soit non bloquant.
- Faites en sorte que le téléchargement notifie l'utilisateur au début et à la fin, mette à jour l'affichage du progrès et ne provoque pas de bug imprévu (notamment liés au parallélisme).
- Mettez à jour les niveaux précédents en notifiant l'utilisateur en cas d'erreur.
- Testez en mettant des fichiers de thème sur `seafile.unistra.fr` et aussi en passant les liens obtenus sur `bitly.com`. Pensez à traiter les erreurs de redirection.

Niveau 8 Styles, icônes et design (4 points)

Objectifs d'apprentissage

- Concevoir le design d'une interface de façon à améliorer son ergonomie en s'aidant de l'état de l'art en interactions Homme-Machine
- Créer des icônes pour une application graphique en Qt en s'aidant de ressources d'icônes
- Construire le style d'une application graphique en Qt

Pour les très grosses applications, les icônes et le style peuvent être créés par des artistes spécialisés. Cependant, un développeur peut tout à fait être amené à s'occuper soi-même d'une partie de ce travail dans des projets plus petits, internes, secrets, ou simplement si les-dits artistes ne sont pas une disponibilité.

Pour les icônes :

- Vous prendrez des icônes de `feathericons.com` (qui permet de modifier leur couleur) ou de `jam-icons.com`, mais pas les deux. Vous pouvez les modifier si vous voulez, par exemple avec *inkscape*. Vous pouvez aussi les créer vous même, mais restez dans le *flat design* pour les icônes.
- Mettez des icônes partout où c'est bien d'en mettre. Il faut que les icônes ne laissent aucun doute sur leur rôle, ou que des informations supplémentaires nécessaires les accompagnent.
- Faites également en sorte que la vue du thème ait un icône. L'icône d'un certain thème doit représenter ce thème d'après son contenu.

Pour le style :

- Utilisez des *stylesheets*.
- Qt donne les styles disponibles pour chaque type de widget : `https://doc.qt.io/qt-5/stylesheet-reference.html`
- Utilisez les styles pour représenter des états utiles, comme par exemple l'état "sélectionné", l'état "invalide", etc.

La recherche en ergonomie ou expérience utilisateur (*UX*) est un champ pluridisciplinaire très vaste. Les façons optimales de représenter les actions ou les objets sont souvent connues et transmises dans des blogs.

Appliquez les réponses à la question de la forme des bords grâce aux liens suivants :

- `https://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes`
- `https://prototypr.io/post/the-rounded-user-experience`
- `https://uxplanet.org/rounded-or-sharp-corner-buttons-def3977ed7c4`

Améliorez l'interface en relisant le cours et en répondant aux questions qui peuvent être soulevées grâce à votre encadrant ou à un navigateur.

Niveau 9 Pour aller plus loin (2 points)

Ce niveau bonus vous encourage à continuer de travailler les détails du projet pour l'améliorer. Il permet de continuer à travailler ses compétences en fignolant l'interface graphique et ses fonctionnalités. À faire en tout dernier et sans sacrifier les autres projets de l'UFR.

- Faites en sorte que si la couleur d'un fichier est l'interpolation de deux couleurs sources, la couleur est remplacée par l'interpolation des deux cibles correspondantes. (1 pt)
- Faites en sorte de traiter de façon optimale le cas où un thème importé a le même nom qu'un thème existant. (0.5 pt)
- Traduisez l'application (0.5 pt).
- Déployez l'application (0.5 pt).
- Faites en sorte de traiter d'autres formes de couleurs que l'hexadécimal (0.5 pt).
- Implémentez toute autre fonctionnalité intéressante (? pt)

Donnez les améliorations que vous avez implémenté dans votre *README*.

Avant le rendu

Dans le README.md :

- Décrivez le processus d'installation, de compilation, et d'exécution de l'application
- Décrivez les fonctionnalités de l'application, section par section, éventuellement avec des vidéos en .gif pour les plus importantes.
- Mettez en valeur les fonctionnalités ergonomiques programmées dans l'application qui ne sont pas facilement remarquables (par exemple : défilement doux, changement de style selon l'état, icône qui change en fonction de l'état de son bouton sous-jacent, etc).