Documentation Technique et guide pratique du Projet

Djamel, Mouhsine & Soufiani

July 26, 2024

Contents

1 Introduction **3**
- 1.1 Objectif du Projet . . . 3
- 1.2 Contexte . . . 3
- 1.3 Technologies Utilisées . . . 3

2 Installation **3**
- 2.1 Prérequis . . . 3
- 2.2 Installation des Dépendances . . . 3
- 2.3 Instructions d'Installation . . . 3

3 Architecture du Projet **4**
- 3.1 Vue d'Ensemble de l'Architecture . . . 4
- 3.2 Diagrammes d'Architecture . . . 4
- 3.3 Description des Composants . . . 4
- 3.4 Structure du Code . . . 4

4 Guide de Développement **5**
- 4.1 Convention de Codage . . . 5
- 4.2 Utilisation des Outils de Développement . . . 5

5 Guide de l'Utilisateur **5**
- 5.1 Description des Fonctionnalités . . . 5
- 5.2 Utilisation de l'Interface Utilisateur . . . 5
- 5.3 Exemple d'Utilisation . . . 6

6 API et Interfaces **6**
- 6.1 Description des API . . . 6
- 6.2 Interfaces Publiques . . . 6
- 6.3 Exemples de Requêtes et Réponses . . . 6

7 Gestion des Erreurs et Débogage **6**
7.1 Messages d'Erreur Courants 6
7.2 Stratégies de Débogage 6
7.3 Gestion des Exceptions 6

8 Tests **6**
8.1 Stratégie de Test 6
8.2 Tests Unitaires 7
8.3 Tests d'Intégration 7
8.4 Perspective et Note aux Futurs Développeurs 7

9 Déploiement **7**
9.1 Instructions de Déploiement 7
9.2 Configuration de l'Environnement 7
9.3 Gestion des Versions 8

10 Contributions **8**
10.1 Instructions pour Contribuer 8
10.2 Code de Conduite 8
10.3 Processus de Revue de Code 8

1 Introduction

1.1 Objectif du Projet

Le projet est une application de dessin qui permet aux utilisateurs de créer et de modifier des dessins en utilisant des formes géométriques et des outils de dessin. L'application inclut des fonctionnalités pour ouvrir, sauvegarder et manipuler des fichiers de dessin.

1.2 Contexte

L'application est développée en utilisant Qt, un framework multiplateforme pour le développement d'applications graphiques. Elle est conçue pour fonctionner sur les systèmes d'exploitation Windows, Linux et macOS.

1.3 Technologies Utilisées

- **Qt Framework** : pour la création de l'interface utilisateur et la gestion des événements.
- **C++** : langage de programmation principal.
- **QFileDialog** : pour les opérations de fichier.
- **QColorDialog** : pour la sélection de couleurs.

2 Installation

2.1 Prérequis

- Qt 5.12 ou ultérieur
- Un compilateur C++ compatible avec Qt

2.2 Installation des Dépendances

1. Téléchargez et installez Qt depuis `https://www.qt.io/download`.
2. Assurez-vous que votre compilateur C++ est configuré avec Qt.

2.3 Instructions d'Installation

1. Clonez le dépôt du projet depuis le dépôt Git :

   ```
   git clone https://github.com/Cmouhsine/PainterApp_ASC
   ```

2. Ouvrez le projet avec Qt Creator.
3. Configurez et construisez le projet en utilisant les outils intégrés de Qt Creator.

3 Architecture du Projet

3.1 Vue d'Ensemble de l'Architecture

Le projet est divisé en plusieurs modules principaux :

- **MenuManager** : Gère les menus et les actions.
- **ToolBarManager** : Gère la barre d'outils et les actions associées.
- **MainWindow** : Contient la fenêtre principale et gère l'interaction entre les différents modules.

3.2 Diagrammes d'Architecture

Incluez des diagrammes de classe, de séquence, ou d'autres types pertinents pour représenter les interactions et la structure du système.

3.3 Description des Composants

- **MenuManager** : Configure les menus de l'application et leurs actions associées.
- **ToolBarManager** : Configure la barre d'outils pour les outils de dessin.
- **MainWindow** : La fenêtre principale qui orchestre les interactions entre les autres composants.

3.4 Structure du Code

Le projet est organisé de la manière suivante :

- **CMakeLists.txt** : Fichier de configuration pour la construction du projet avec CMake.
- **PainPrj/** : Répertoire principal contenant les fichiers d'en-tête et les fichiers source.
  - **Header Files/** : Contient les fichiers d'en-tête (`.h`).
    - **copypaste.h** : Déclarations pour la gestion des opérations de copier-coller.
    - **dragbutton.h** : Déclarations pour les boutons de glisser-déposer.
    - **filehandler.h** : Déclarations pour la gestion des fichiers.
    - **mainwindow.h** : Déclarations pour la fenêtre principale de l'application.
    - **paintcanvas.h** : Déclarations pour le canevas de dessin.
    - **redoundo.h** : Déclarations pour la gestion des opérations de redo/undo.

  - **Source Files/** : Contient les fichiers source (`.cpp`).
    * **copypaste.cpp** : Implémentation des fonctionnalités de copier-coller.
    * **dragbutton.cpp** : Implémentation des boutons de glisser-déposer.
    * **filehandler.cpp** : Implémentation de la gestion des fichiers.
    * **main.cpp** : Point d'entrée de l'application.
    * **mainwindow.cpp** : Implémentation de la fenêtre principale.
    * **paintcanvas.cpp** : Implémentation du canevas de dessin.
    * **redoundo.cpp** : Implémentation des fonctionnalités de redo/undo.

- **mainwindow.ui** : Fichier de conception de l'interface utilisateur pour la fenêtre principale.

4 Guide de Développement

4.1 Convention de Codage

- Utilisez des noms de variables descriptifs.
- Suivez les conventions de codage C++ et Qt.
- Documentez le code avec des commentaires clairs.

4.2 Utilisation des Outils de Développement

- **Qt Creator** : IDE principal utilisé pour le développement et le débogage.
- **GCC/Clang/MSVC** : Compilateurs pour la construction du projet.

5 Guide de l'Utilisateur

5.1 Description des Fonctionnalités

- **Création de Projet** : Permet de commencer un nouveau dessin.
- **Ouverture de Fichier** : Charge un dessin existant depuis un fichier.
- **Sauvegarde de Fichier** : Enregistre le dessin actuel dans un fichier.

5.2 Utilisation de l'Interface Utilisateur

Décrivez les éléments de l'interface utilisateur et leur utilisation, comme les menus, la barre d'outils, et les panneaux.

5.3 Exemple d'Utilisation

- **Créer un nouveau dessin** : Cliquez sur "Fichier" -¿ "Nouveau".
- **Choisir une couleur** : Cliquez sur "Outils" -¿ "Palette de couleurs".

6 API et Interfaces

6.1 Description des API

Décrivez les interfaces publiques disponibles pour les autres développeurs, y compris les signatures de méthodes et les paramètres.

6.2 Interfaces Publiques

- **MenuManager** : `createNewProject()`, `openFile()`, `saveFile()`
- **ToolBarManager** : `getToolBar()`, `addAction()`

6.3 Exemples de Requêtes et Réponses

Montrez des exemples d'appels de méthode et de leurs résultats.

7 Gestion des Erreurs et Débogage

7.1 Messages d'Erreur Courants

- **Erreur de fichier** : "Impossible d'ouvrir le fichier pour la lecture/écriture."
- **Erreur de sélection de couleur** : "Aucune couleur sélectionnée."

7.2 Stratégies de Débogage

Utilisez les outils de débogage intégrés de Qt Creator pour suivre les erreurs et les exceptions.

7.3 Gestion des Exceptions

Assurez-vous que toutes les exceptions sont correctement capturées et gérées pour éviter les plantages de l'application.

8 Tests

8.1 Stratégie de Test

Initialement, nous avions prévu de réaliser des tests unitaires et d'intégration pour assurer la fiabilité et la robustesse de notre application. Cependant, en

raison du manque de temps, nous n'avons pas pu mener ces tests à terme. Nous encourageons les futurs développeurs à poursuivre cet effort pour améliorer la qualité du code.

8.2 Tests Unitaires

Les tests unitaires devraient être effectués pour chaque fonction clé du code. Utilisez des frameworks de test comme `QTest` pour automatiser ces tests. Voici quelques zones spécifiques à tester :

- Fonctionnalités de copier-coller.
- Gestion des fichiers (ouverture, sauvegarde).
- Opérations de dessin (lignes, rectangles, ellipses).
- Fonctionnalités redo/undo.

8.3 Tests d'Intégration

Les tests d'intégration devraient assurer que les différents composants fonctionnent correctement ensemble. Cela inclut des tests comme :

- Interaction entre le canevas de dessin et les outils de la barre d'outils.
- Gestion des événements de l'interface utilisateur (clics, déplacements de souris).
- Sauvegarde et chargement de fichiers de dessin avec tous les éléments correctement restaurés.

8.4 Perspective et Note aux Futurs Développeurs

En raison des contraintes de temps, les tests n'ont pas été complètement réalisés. Nous recommandons fortement aux futurs développeurs de mettre en œuvre et d'améliorer la suite de tests pour garantir une couverture complète et une robustesse accrue du projet. La mise en place d'une stratégie de test rigoureuse permettra de prévenir les régressions et d'améliorer la maintenabilité du code.

9 Déploiement

9.1 Instructions de Déploiement

Décrivez les étapes pour déployer l'application sur différentes plateformes.

9.2 Configuration de l'Environnement

Incluez les paramètres et configurations nécessaires pour l'exécution de l'application.

9.3 Gestion des Versions

Décrivez le système de gestion des versions utilisé (par exemple, Git) et comment créer des versions.

10 Contributions

10.1 Instructions pour Contribuer

Expliquez comment les développeurs peuvent contribuer au projet, y compris le processus de soumission de pull requests.

10.2 Code de Conduite

Incluez un code de conduite pour les contributeurs afin d'encourager des interactions respectueuses.

10.3 Processus de Revue de Code

Décrivez le processus de revue de code, y compris les critères d'acceptation des contributions.