Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain

Compte Rendu Projet C++

Encadres par : Pr. Ikram Ben Abdel Ouahab

Réalisé par : Khabali Ayoub (P130447036)

Sommaire :

- Introduction 2
- Objectif du projet 2
- Structure de projet 5
- Conception et Modélisation..... 7
- Fonctionnalités Principales 11
- Les contraintes 11
- Conclusion 11
- Perspectives d'émelioration 12
- Bibliographique 12

Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain

Introduction :

Ce rapport présente le projet de création d'un jeu de cartes en C++ en utilisant le framework Qt. Le jeu de cartes choisi est "RONDA". Le projet a été développé dans le cadre d'un cours universitaire visant à approfondir les compétences en programmation C++ et à explorer l'utilisation du framework Qt.

Objectif du Projet :

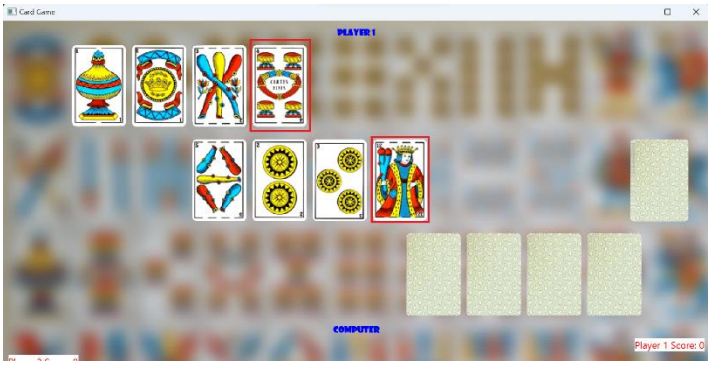
L'objectif principal du projet était de concevoir et de mettre en œuvre un jeu de cartes fonctionnel en utilisant le langage de programmation C++ et le framework Qt. Le jeu de cartes "RONDA" a été choisi pour sa popularité et son attrait culturel.

Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain

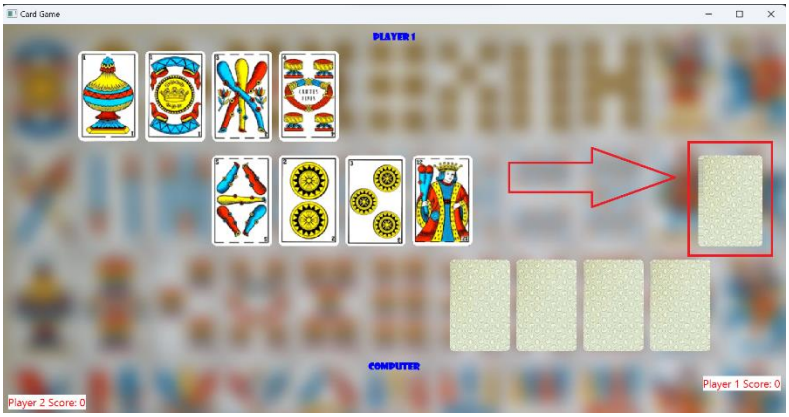
Structure du Projet :

Le projet est organisé en plusieurs fichiers source, chacun responsable d'une partie spécifique du jeu. Les fichiers principaux inclus dans le projet sont :

- card.h / card.cpp : Définition et implémentation de la classe Card représentant une carte du jeu.

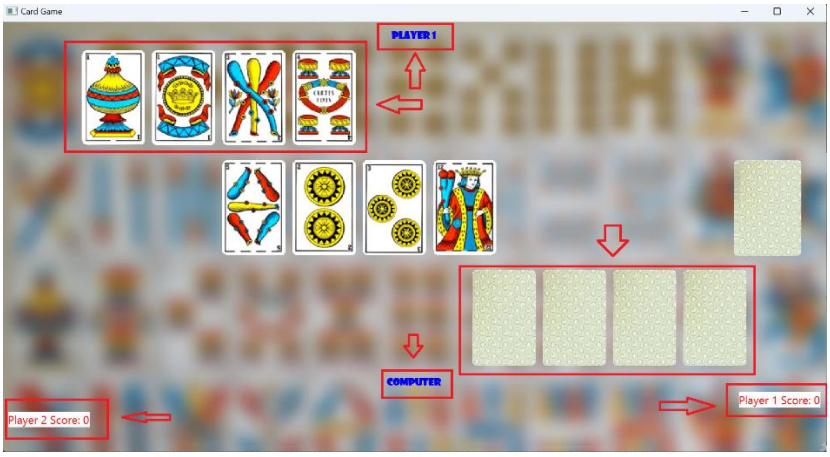


- deckofcards.h / deckofcards.cpp : Définition et implémentation de la classe DeckOfCards gérant le paquet de cartes.



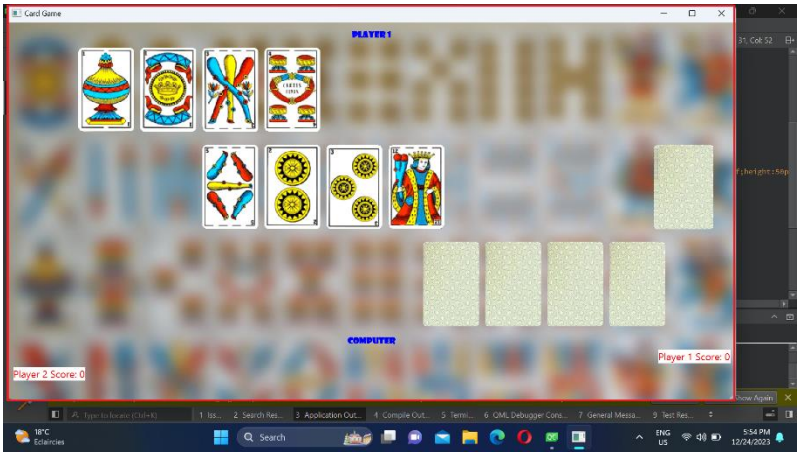
- player.h / player.cpp : Définition et implémentation de la classe Player représentant un joueur et ses actions.

Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain



- clickablepixmapitem.h / clickablepixmapitem.cpp : Définition et implémentation de la classe ClickablePixmapItem pour rendre les cartes cliquables.

- cardspage.h / cardspage.cpp : Définition et implémentation de la classe CardsPage, la page principale du jeu, utilisant les classes précédentes.



- starterpage.h / starterpage.cpp : Définition et implémentation de la classe StarterPage, la page d'accueil du jeu.

Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain

- mainwindow.h / mainwindow.cpp : Définition et implémentation de la classe MainWindow, la fenêtre principale de l'application.

Conception et Modélisation :

1. Définition des Entités du Jeu :

Joueur (Player) :

Attributs : nom, main (liste de cartes), score, nombre de cartes en main.

Méthodes : jouer une carte, mettre à jour le score, piocher des cartes initiales.

Carte (Card) :

Attributs : valeur, image.

Méthodes : comparer avec d'autres cartes.

Jeu Central (Ka3a) :

Attributs : liste de cartes.

Méthodes : initialiser le jeu, vérifier les correspondances, mettre à jour le jeu après un coup.

Deck de Cartes (CardDeck) :

Attributs : liste de cartes restantes.

Méthodes : piocher une carte.

2. Définition des Actions du Jeu :

Jouer une Carte :

Le joueur joue une carte de sa main.

```
void Player::playCard(const Card& card)
{
    auto it = std::find(hand.begin(), hand.end(), card);
    if (it != hand.end())
    {
        hand.erase(it);
    }
}
```

La correspondance est vérifiée avec les cartes dans le jeu central.

```
matchingCards.clear();
// Check if the played card matches any card in ka3a
for (auto it = ka3a.begin(); it != ka3a.end(); ) {
    const Card& ka3aCard = *it;

    if (ka3aCard.getValue() == card.getValue()) {
        matchingCards.append(ka3aCard);
        it = ka3a.erase(it);                                                    ⚠ Mixing iterators with const_ite

        // Keep incrementing and checking for consecutive values
        int nextValue = card.getValue() + 1;
        while (true) {
            auto nextValueCardIt = std::find_if(ka3a.begin(), ka3a.end(),
                                                [nextValue](const Card& c) { return c.getValue() == nextValue; });

            if (nextValueCardIt != ka3a.end()) {
                matchingCards.append(*nextValueCardIt);
                it = ka3a.erase(nextValueCardIt);                               ⚠ Mixing iterators with const_ite
                nextValue++;
            } else {
                break;
            }
        }
    } else {
        ++it;
    }
}

if (matchingCards.isEmpty()) {
    ka3a.append(card);
}
player1.handSize--;

// Update the player's earning based on the number of removed cards (including the played card)
if (!matchingCards.isEmpty())
player1.updateEarning(matchingCards.size()+1);
```

Le jeu central est mis à jour en conséquence.

```
// Update the display to reflect the ch
displayPlayerCards(player, true);
ka3aScene->clear();
displayCards(ka3aScene, ka3a);
matchingCards.clear();

// If the current player is player1, le
```

Piocher une Carte :

Le joueur pioche une carte du deck de cartes.

La carte est ajoutée à la main du joueur.

```
15
16  Card DeckOfCards::drawCard()
17  {
18
19
20      // Randomly select a card index
21      int index = QRandomGenerator::global()->bounded(deck.size());
22      Card drawnCard = deck.takeAt(index);
23      return drawnCard;
24  }
25
26
```

Jouer une Carte Aléa toire (pour l'ordinateur) :

```
void CardsPage::playRandomCard(QList<Card>& ka3a, Player& player)
{
    int difficulty = selectedDifficulty;

    int randomIndex;
    QList<Card> matchingCards;

    if (difficulty == 1) {
        randomIndex = rand() % player.getHand().size();
    }
```

```
    Card randomCard = player.getHand()[randomIndex];    ⚠1
    player.playCard(randomCard);
```

3. Interface Utilisateur (UI) :

Affichage des cartes des joueurs.

Affichage du jeu central (Ka3a).

Affichage du deck de cartes.

Scores des joueurs.

4. Interaction Utilisateur :

Clic sur une carte pour la jouer.

Bouton pour terminer le tour.

5. Cycle de Jeu :

Les joueurs jouent à tour de rôle.

Le jeu central est mis à jour après chaque coup.

Le jeu se termine lorsque les conditions de fin sont remplies.

Fonctionnalités Principales :

Le jeu implémente les fonctionnalités suivantes :

- Affichage d'une page d'accueil avec des options de jeu, de règles et de sortie.

- Sélection de la difficulté du jeu avant de commencer.

Universite Abdelmalek Essaadi
Faculte des sciences et techniques de Tanger
Département Génie Informatique
Cycle licence : IDAI
Pr. Ikram Ben Abdel Ouahab
Jeu de Cartes Marocain

Distribution de cartes aux joueurs et à la "ka3a" (tas central) :

La fonction initializeKa3a efface les cartes existantes dans le jeu central (ka3a) et ajoute ensuite 4 cartes uniques avec des valeurs distinctes à partir du paquet principal de cartes. Elle garantit que le jeu central est prêt avec des cartes uniques pour une nouvelle partie.

```
327
328 void CardsPage::initializeKa3a()
329 {
330     // Clear existing cards in ka3a
331     ka3a.clear();
332
333     // Keep track of unique values
334     QSet<int> uniqueValues;
335
336     // Add 4 cards with unique values
337     while (ka3a.size() < 4) {
338         Card newCard = deckOfCards.drawCard();
339         int cardValue = newCard.getValue();
340
341         // Check if the value is unique
342         if (!uniqueValues.contains(cardValue)) {
343             ka3a.append(newCard);
344
345             // Mark the value as used
346             uniqueValues.insert(cardValue);
347         } else {
348             deckOfCards.getDeck().append(newCard);
349         }
350
351     }
352
353
```

- Mécanisme de jeu où les joueurs jouent des cartes et marquent des points :
    - Au début de la partie, 4 cartes sont retournées sur le tapis.

- A chaque tour, 4 cartes sont distribuées par joueur. A tour de rôle, les joueurs pose une carte sur le tapis. si une paire est formée le joueur récolte les 2 cartes ainsi que les cartes supérieures qui font une suite.
- Par exemple, si le tapis contient 1, 2, 3 et 5, en jouant la carte 1, le joueur remporte la paire de 1, la 2 et la 3 soit 4 cartes.

- Logique de jeu pour l'ordinateur (joueur automatique) avec différentes difficultés.

  La logique de chaque niveau :

  Niveau 1 : Facile

  - En mode facile, l'ordinateur (adversaire) joue une carte au hasard parmi celles de sa main.

  Niveau 2 : Moyen

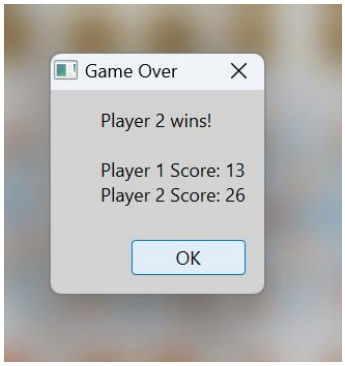
  - En mode moyen, l'ordinateur vérifie s'il y a une carte correspondante dans le jeu central (ka3a).
  - Si une carte correspondante est trouvée, l'ordinateur joue cette carte correspondante.
  - Si aucune carte correspondante n'est trouvée, il joue une carte au hasard.

  Niveau 3 : Difficile

  - En mode difficile, l'ordinateur vérifie s'il y a une carte avec la même valeur dans le jeu central (ka3a).
  - Si une carte correspondante est trouvée, l'ordinateur joue cette carte correspondante.
  - Si aucune carte correspondante n'est trouvée dans la main du joueur, il joue une carte qui n'existe pas dans la main du joueur.

- Affichage des scores des joueurs.

- Détection de la fin du jeu et affichage du gagnant.



Les Contraintes :

- Les difficultés auxquelles j'ai été confronté lors de la programmation de ce projet résident dans le manque de ressources et de documentation de la dernière version du framework Qt (6.6). Les fonctions du framework sont en constante évolution à chaque version, ce qui m'a obligé à m'en tenir à la documentation publiée par les développeurs du framework et à travailler en fonction de celle-ci.

- De plus, les difficultés ont émergé lors de la mise en place de la fonctionnalité permettant de cliquer sur les cartes. En effet, la classe QPixmap, responsable de l'affichage des images, ne prend pas en charge cet événement. Par conséquent, j'ai dû créer une classe personnalisée appelée ClickablePixmapItem afin de rendre cet événement possible.

Conclusion :

En conclusion, le projet a été une opportunité enrichissante pour appliquer les concepts de la programmation orientée objet en C++ et explorer le développement d'applications graphiques avec le framework Qt. Le jeu de cartes fonctionne comme prévu, offrant une expérience interactive aux utilisateurs.

Perspectives d'émelioration :

Des améliorations futures pourraient inclure l'ajout de fonctionnalités avancées telles que des animations, des effets sonores, et la possibilité de sauvegarder et de charger des parties. De plus, une amélioration de l'interface utilisateur pourrait rendre l'expérience de jeu encore plus attrayante.

Ce projet a permis d'acquérir des compétences pratiques dans le développement d'applications C++ avec le framework Qt, tout en offrant une expérience ludique grâce à la création d'un jeu de cartes complet.

Bibiliographique :

ayoubkhabali/Ronda: le projet de création d'un jeu de cartes en C++ en utilisant le framework Qt. Le jeu de cartes choisi est "RONDA". Le projet a été développé dans le cadre d'un cours universitaire visant à approfondir les compétences en programmation C++ et à explorer l'utilisation du framework Qt. (github.com)