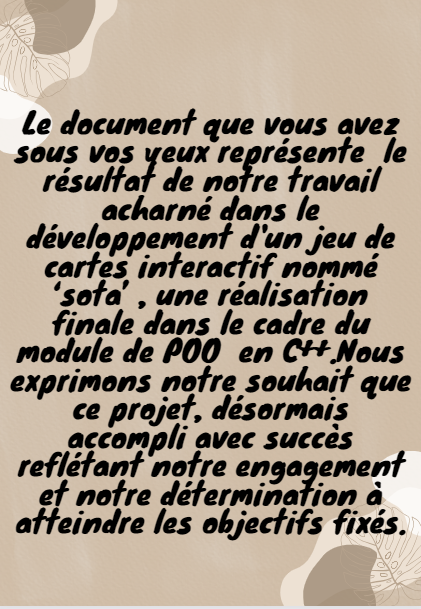
**Rapport de jeu ****Définition**

Sota est un jeu de carte marocain qui contient deux joueurs, on a 40 cartes, ensuite on distribue les cartes aux joueurs aléatoirement au moitie, puis les deux joueurs commencent à jouer par tour, chacun pose une carte au centre de l’interface, et ainsi de suite jusqu’à ce qu’il reste un joueur possédant des cartes.

**Règles de jeu :**

Nombre de joueurs : le jeu se joue généralement à deux joueurs.

Le but de jeu : c’est d’éliminer toutes les cartes d’un joueur.

Règles de jeu : un joueur pose une carte au milieu, si l’adversaire a une carte de même valeur, il la pose, sinon il va poser une carte qui a une valeur inférieur ou supérieur à celle de la carte posée.

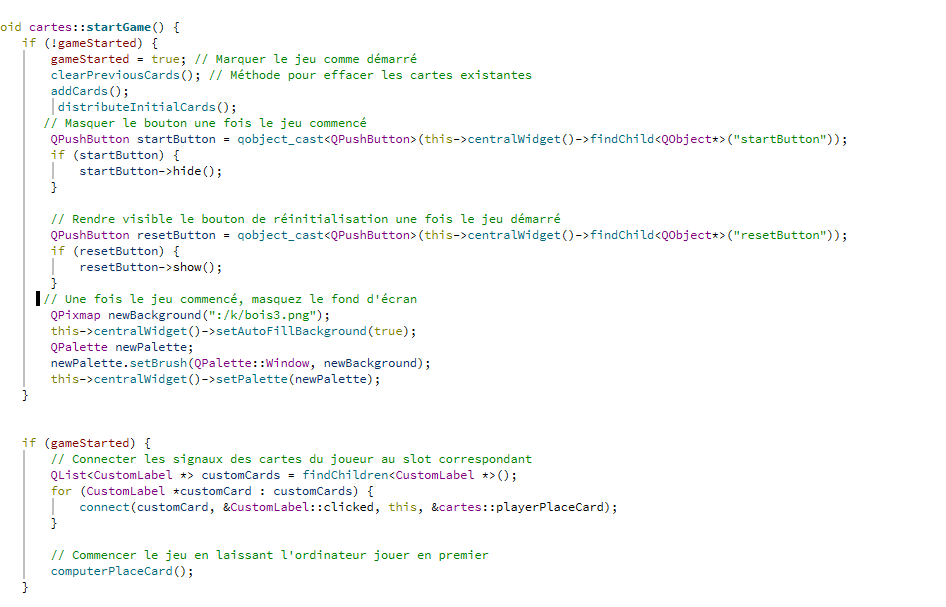
S’il n’a pas, l’autre joueur va jouer encore une fois, est ainsi de suite.et le gagnant est celui qui a posé le plus grand nombre de cartes

Détermination de fin de jeu : on va vérifier le nombre de cartes posés, si un joueur est débarrassé de tous ses cartes alors le jeu est terminé.

Détermination de gagnant : celui qui est débarrassé de tous ses cartes le premier sera le gagnant et l’autre sera le perdant.

**Déroulement :**

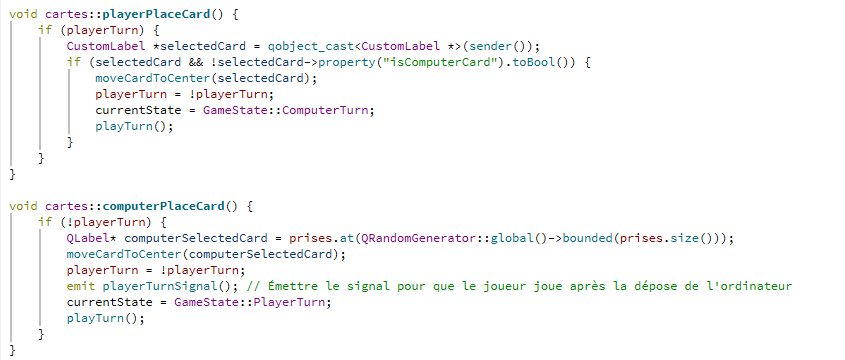
* **Lancement de jeu :**  
  Pour commencer le jeu on a basé sur la fonction **startGame()** pour lancer le jeu, cette fonction est démarrée lorsque le bouton «  commencer le jeu «  est appelé, et démarre le tour de joueur en appelant la fonction **playTurn().**



* **Distribution des cartes :**  
  pour distribuer les cartes on a utilisé la fonction **distributeInitialCards()**.Cette fonction distribue initialement les cartes entre le joueur et l’ordinateur, elle utilise la variable **prises** qui contient les cartes mélangés. Elle appelle également la fonction **centerCards()** pour positionner les cartes sur l’interface.
* **Gestion des cliques :**

Pour gérer les cliques sur les cartes on a utilisé deux fonctions.

* La première c’est **playerPlaceCard().** Cette fonction lorsqu’une carte de joueur est cliquée, si c’est le tour du joueur il déplace la carte cliquée vers le centre met à jour le tour et appelle **playTurn().**
* La deuxième fonction est **computerPlaceCard()** cette fonction est appelle lors le tour de l’ordinateur ,elle choisit aléatoirement une carte de l’ordinateur et la déplace vers le centre, ensuite elle émet le signal **playerTurnSignal()** pour que le joueur joue.



* **Gestion de déplacement des cartes :**

On a défini cette fonction **moveCardToCenter(Qlabel \*Card)** pour déplacer la carte spécifiée vers le centre de l’interface en utilisant une animation.



1. Int cardwidth = 80, int cardHeight = 100 :

Ces lignes déclarent les dimensions standard, d’une carte en largeur et en hauteur, ces valeurs sont utilisées pour définir la taille de la carte ors de son déplacement.

1. Int centerOffset= (width () – cardWidth) /2 :

Calcule le décalage horizontal pour placer la carte au centre de la fenêtre, il soustrait la largeur totale de la fenêtre, divise par deux pour obtenir la marge de chaque côté de la carte.

1. QPropertyAnimation \*moveAniamtion = new QPropertyAnimation(card, “geometry”):

Cela crée une animation sur la propriété ‘geometry’ de l’objet card, ce qui signifie que cette animation déplacera la carte.

1. MoveAnimation->setDuration(1000) :

Fixe la durée de l’animation a 1000 millisecondes (ou 1 seconde).

1. MoveAnimation->setStartValue(card->geometry ()) :

Définit la position de départ de l’animation en utilisant la position actuelle de la carte.

1. MoveAnimation->setEndValue (QRect (centeroffset, height () /2, cardwidth , cardHeight) :

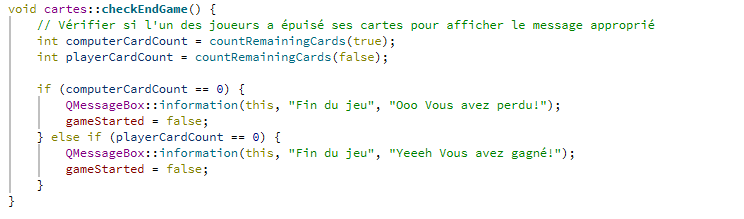
Définit la position finale de l’animation en utilisant les valeurs de centerOfsset pour la position horizontale, et la moitié de la hauteur de la fenêtre pour la position verticale, ainsi que les dimensions de la carte.

1. MoveAnimation->setEasingCurve (QEasingCurve :: InOutQuad) :

Définit la courbe de l’accélération pour l’animation, ici QEasingCurve :: InOutQuad est utilisé pour que la vitesse de l’animation commence lentement , s’accélère au milieu, puis ralentit à la fin.

* **Condition de fin de jeu** :

Pour déterminer la condition de fin de jeu on a utilisé la fonction **checkEndGame()** qui conte le nombre de cartes restants pour le joueur et l’ordinateur . Si le nombre de cartes d’un joueur est égal à zéro, le jeu est considéré comme terminé. Dans ce cas une boite de dialogue **QMessageBox** est affiche pour informer le joueur de résultat de jeu.



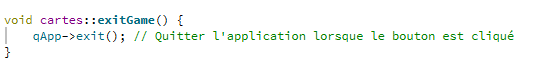
* **Réinitialisation de jeu :**

On a offrir une possibilité de recommencer le jeu par la fonction **resetGame()** qui réinitialise le jeu en masquant le bouton de réinitialisation, en affichant à nouveau le bouton « commencer le jeu », en effaçant les cartes existantes , en ajoutant de nouvelles cartes mélangés et en appelant la fonction **distributeInitialCard().**



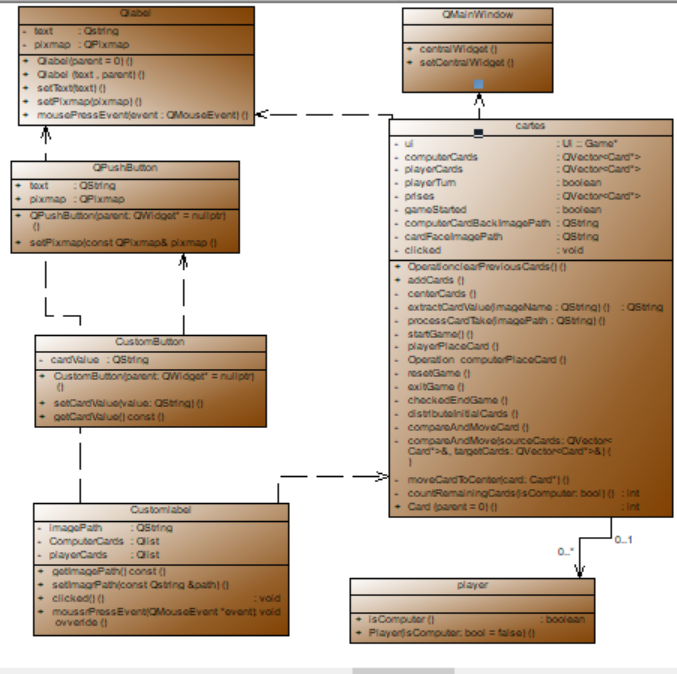
* **Quitter le jeu :**

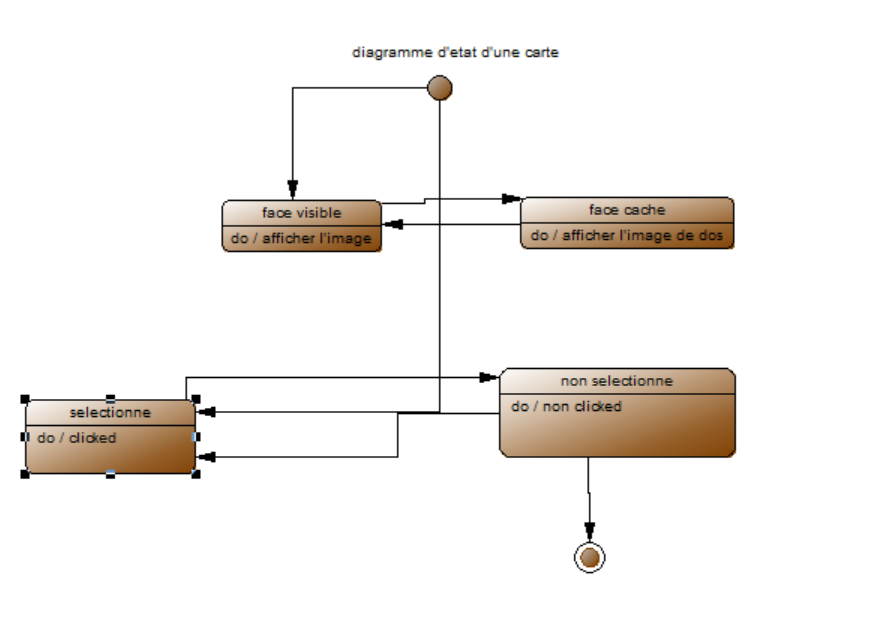
On a ajouté une fonction appelée **exitGame()** pour quitter le jeu en connectant un bouton appelé « Quitter le jeu ».



**Modelisation de projet :**

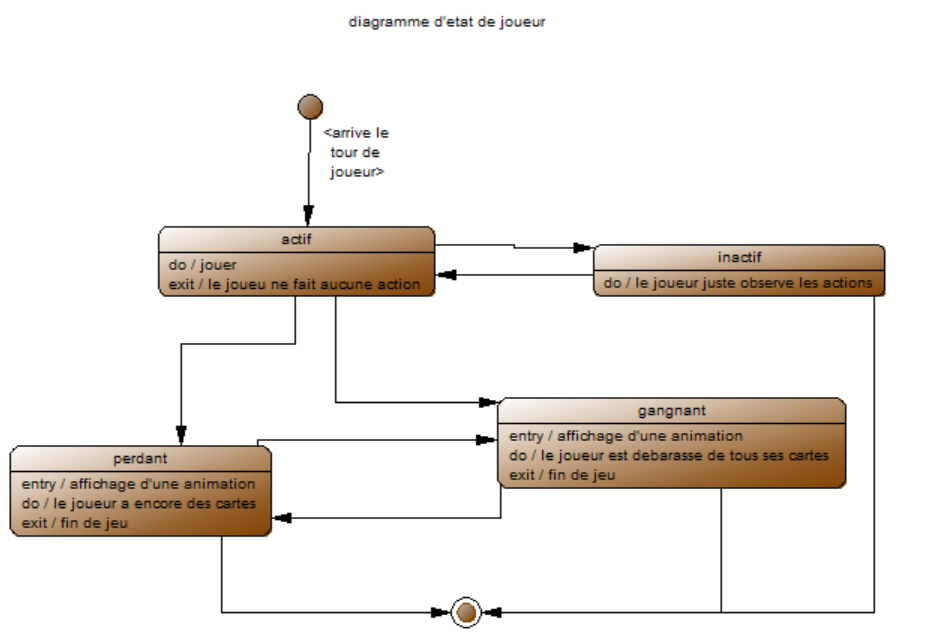
**Diagramme de classes :**



**Diagramme d’etat :** 

Une image contenant texte, capture d’écran, ligne

Description générée automatiquement



**Les difficultés :**

* Parmi les difficultés qu’on rencontré c’est qu’on a défini les fonctions correctement mais, le résultat ne s’affichera pas dans l’interface.
* La deuxième difficulté qu’on a rencontrée c’est que on n’a pas pu retourner les images des cartes a face caches.