كلية العلوم و التقنيات بطنجة
ⵜⴰⵖⵡⵍⴰⵢⵜ ⵏ ⵜⵣⴰⵡⵉⵜ ⴷ ⵜⵙⵙⵏⵜⵉⵏ ⵜⵉⴽⵏⵓⵍⵓⵊⵉⵜⵉⵏ ⵏ ⵟⴰⵏⵊⴰ
Faculté des Sciences et Techniques de Tanger

HEZ 2

MOROCCAN CARD GAME

Developed by :
BELHACHMI ABDELAZIZ

Using:
QT 6 Framework

Supervised by Professor:
Ikram Ben Abdel Ouahab

Introduction

Tthis report unveils the creation of a card game using C++, Qt 6, and QML.

This project merges traditional gaming with modern programming, bringing together robust logic and dynamic user interfaces.

Follow my journey through the codebase, exploring the harmony of C++ and Qt 6, and the power of QML in crafting an engaging card game experience.

This report serves as both documentation and an invitation to discover the synergy of software development and creative design principles.

My Vision

The goal when creating this card game is to bring the joy of traditional card games into the digital world.

I want to offer players a fun and easy-to-play experience using C++, Qt 6, and QML.

By blending old-school card gaming with modern technology,

My aim is to provide a familiar yet fresh gaming experience.

This project is all about making classic games accessible in a new way, showcasing the fun side of coding and game design.

Why Qml And Not QWidgets ?

I chose QML because it's simple, making it easy to design and implement the user interface.

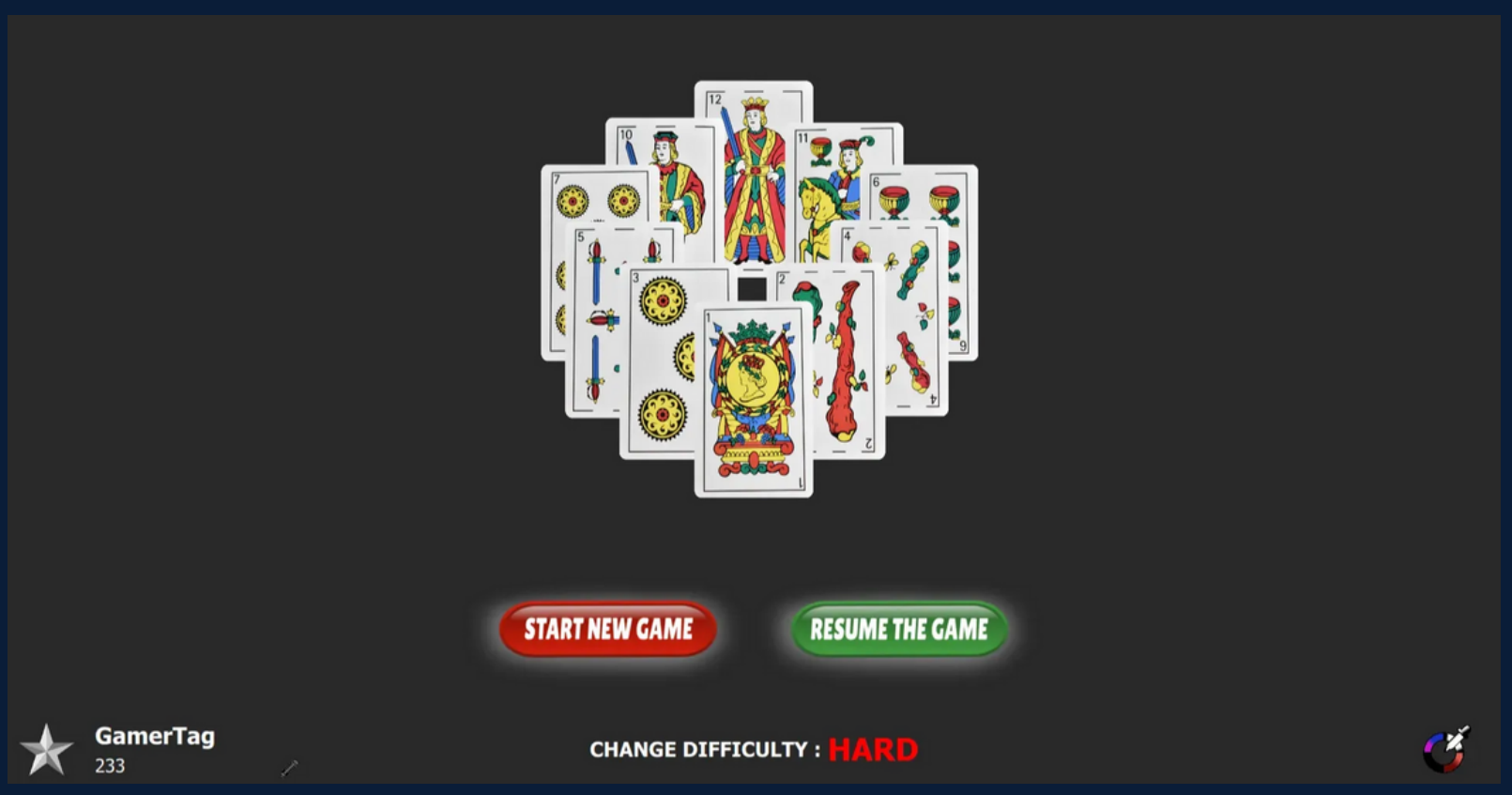
It's really good at handling cards movements and adding beautiful animations and Sound effects.

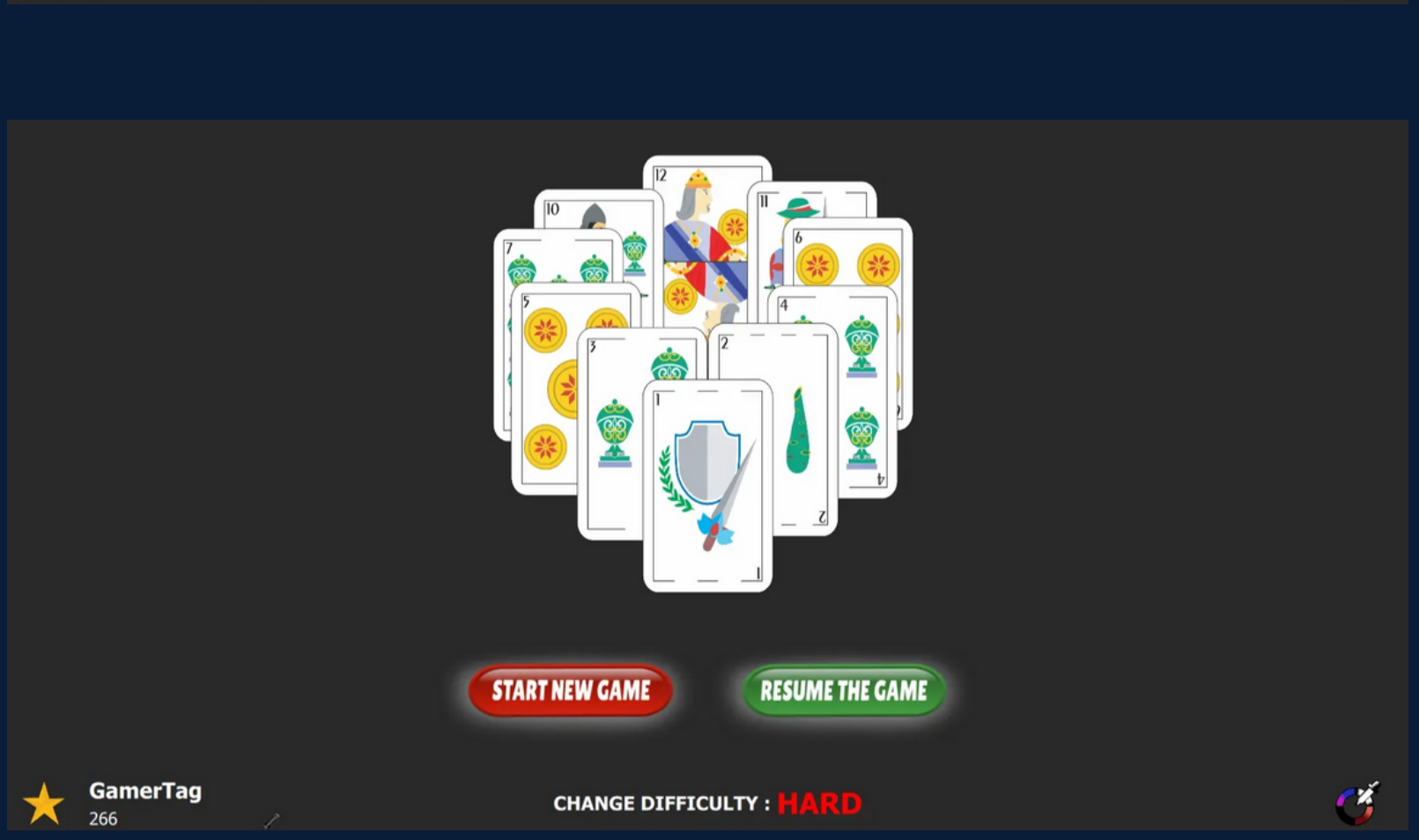
Despite QML's responsibility for the frontend, it's crucial to note that the core game functionality and logic are managed by the well structured C++ code.

It's like QML makes things pretty, and C++ makes sure everything works smoothly behind the scenes.

Game Showcase

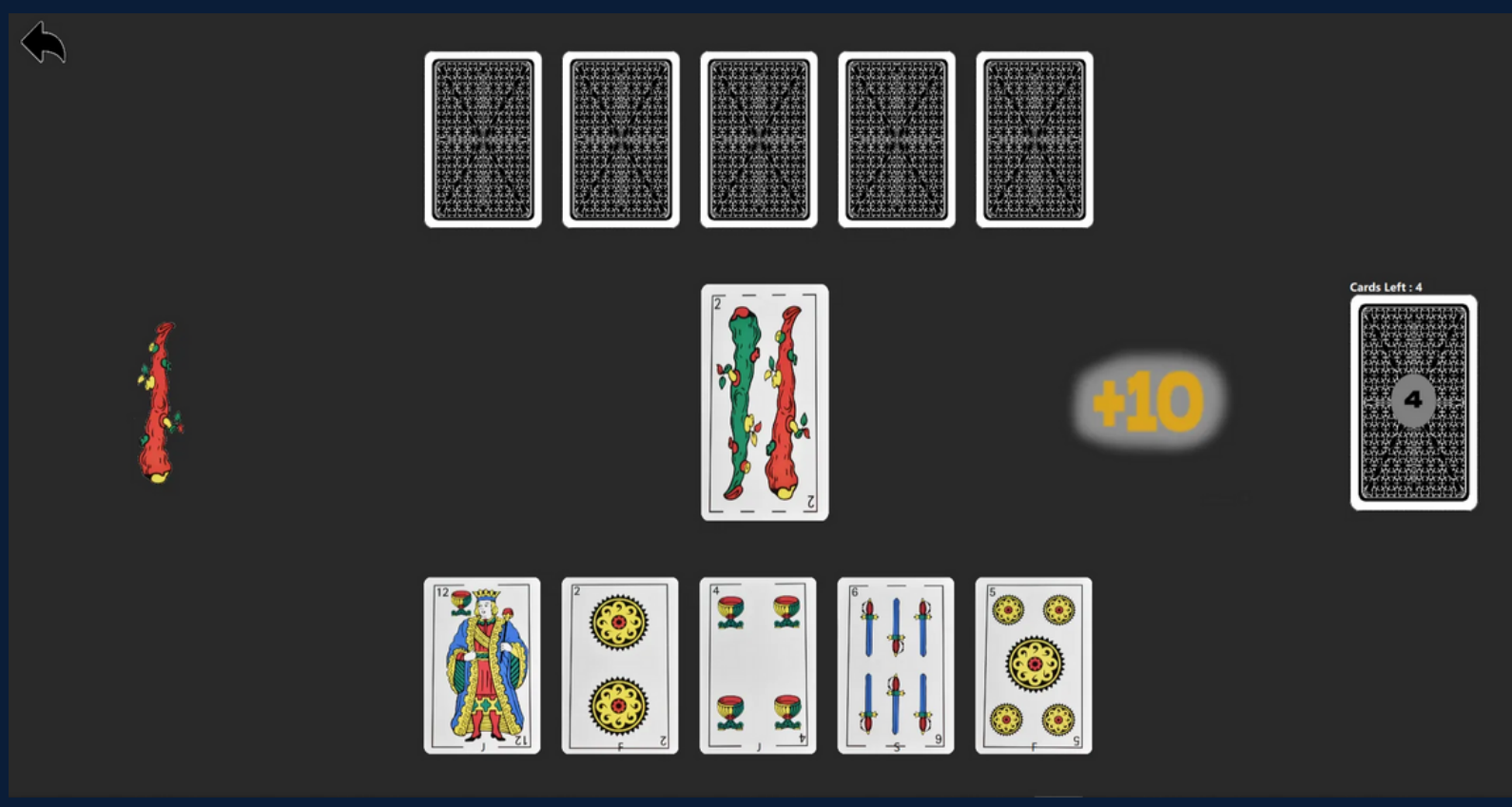
Game Main Menu

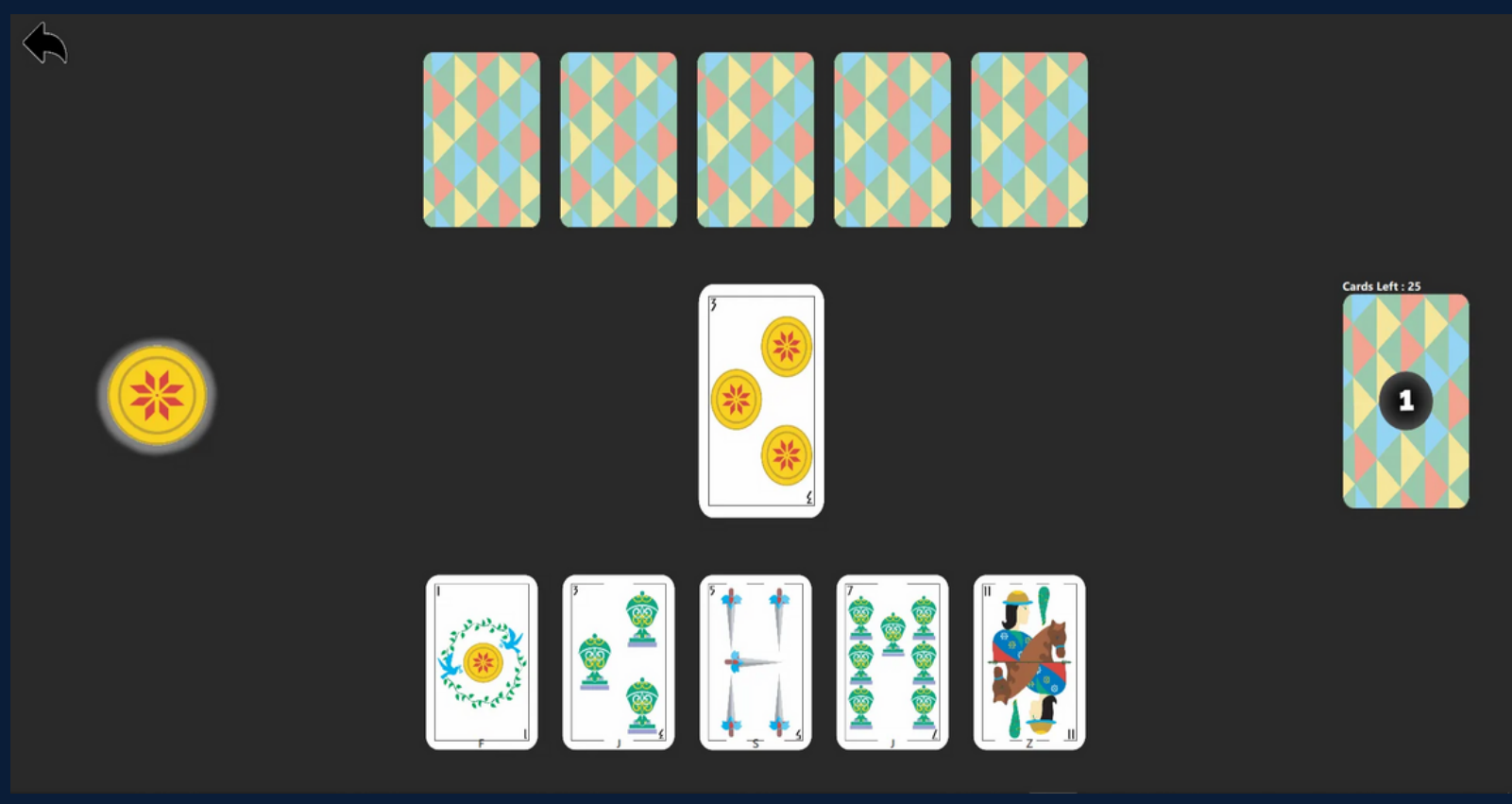




Game Showcase

Game Table



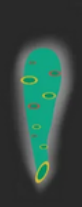


Game Showcase

Request a Card Type

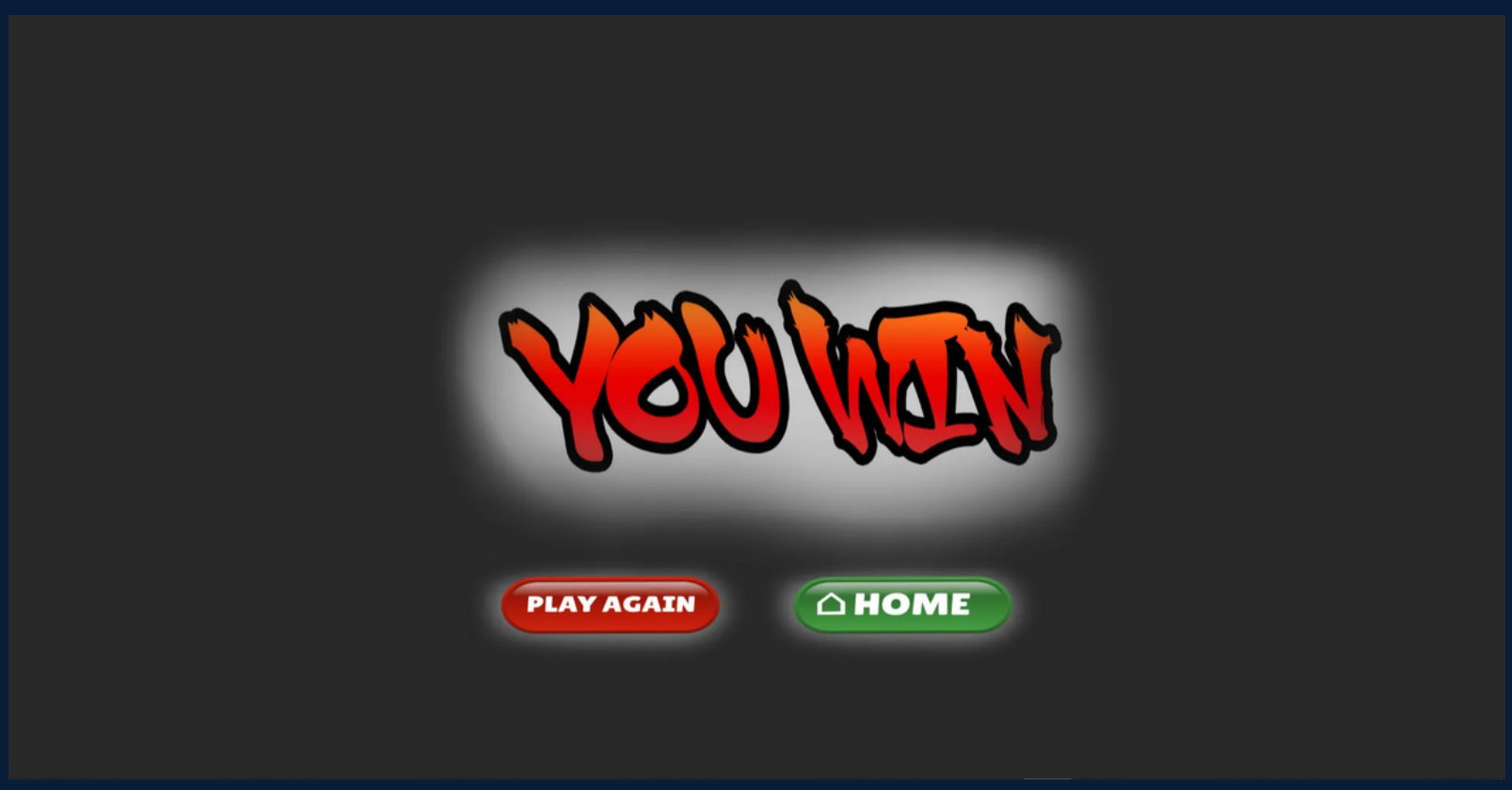


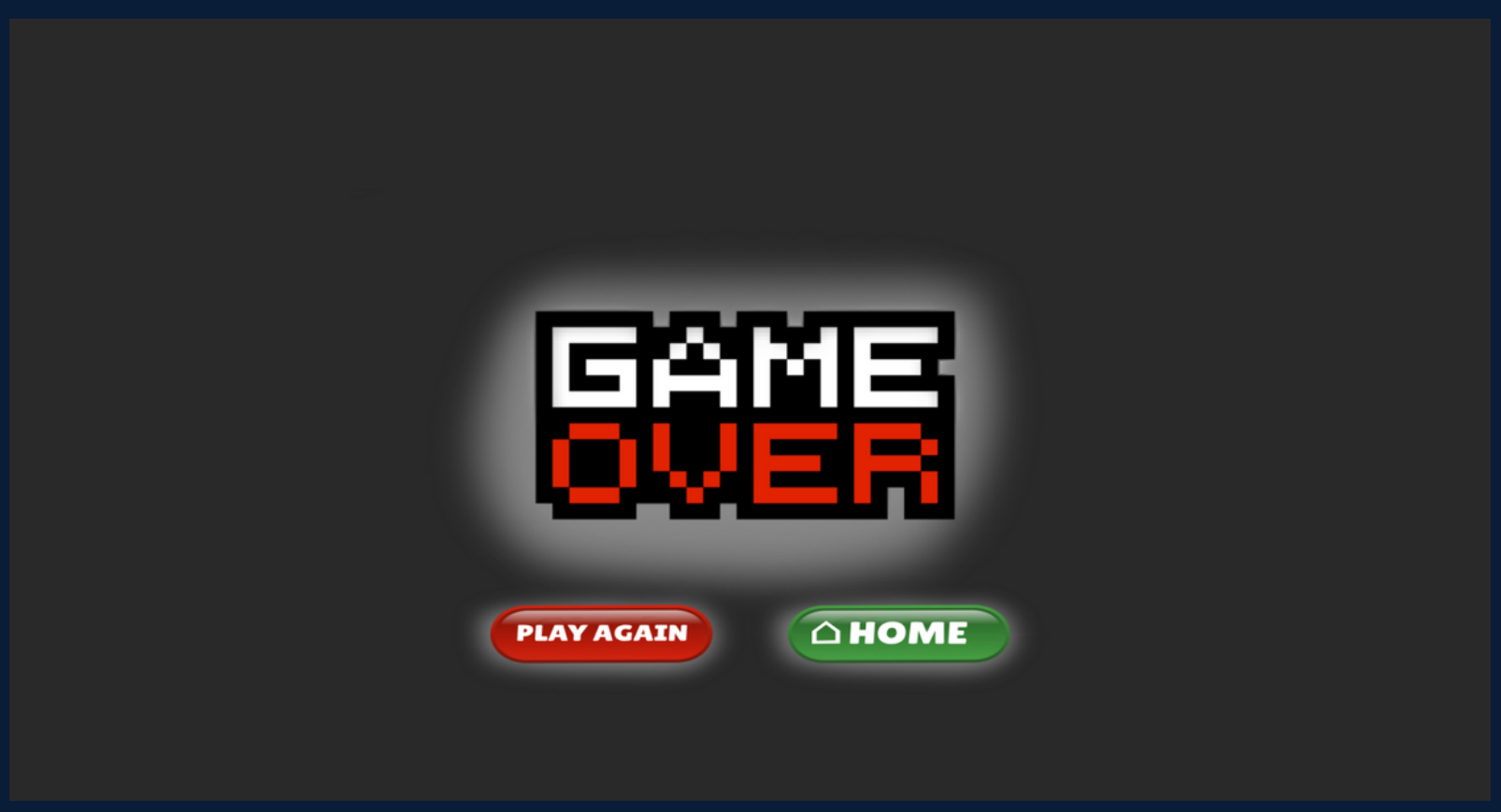


Game Showcase

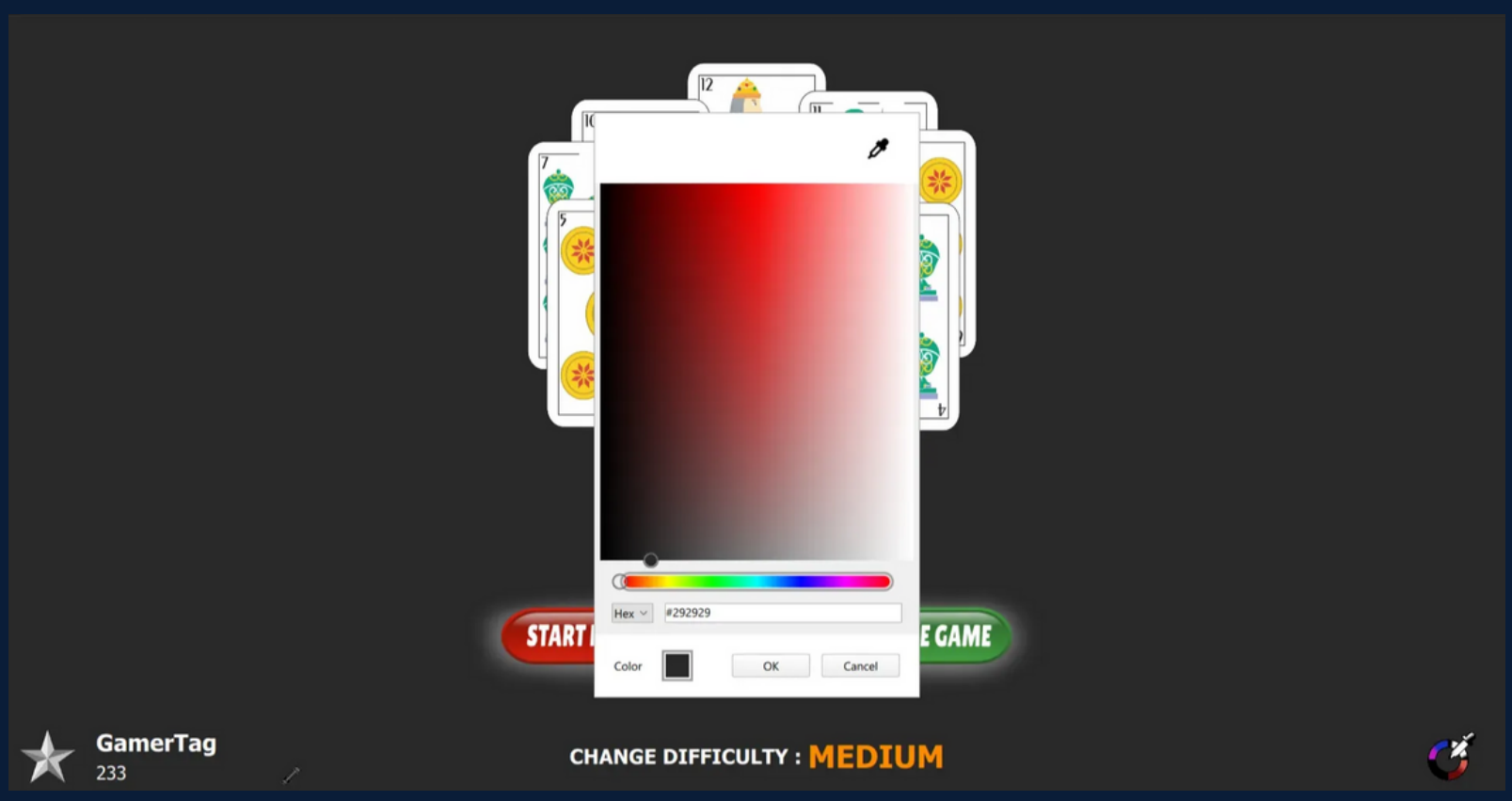
End Game





Game Showcase

Adjustable Background Color



Game Rules

This game rules are easy to understand and to follow , but does not mean it is an easy game

Actually every card you play can seriously shake up the game.

One wrong move can quickly turn things around, leading to a loss.

That's why I specifically chose this game – it keeps you on your toes.

Game Rules

These are the game rules that you got to follow:

1. Put down a card that matches the last one on the table, either by number or type. We got four types: sword, coin, clubs, and cups.

2. Drop a card with the number 1 to make your opponent take a break. You get to play again.

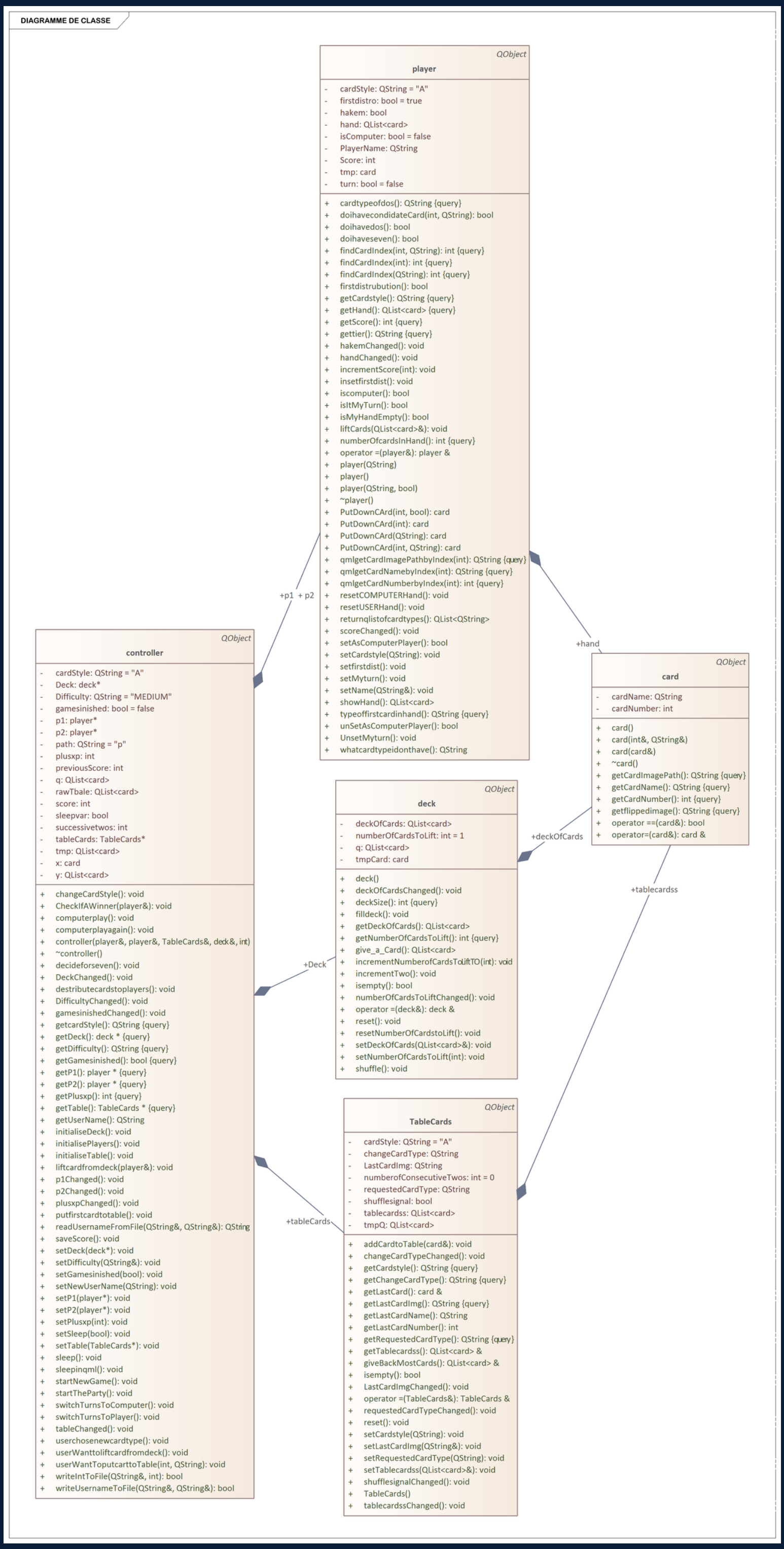
3. Play a card with the number 2, and your opponent has to pick up 2 cards from the deck. If they play a 2, you do the same “lift 2 cards”.

3-a. You have the option to sequentially place cards with the number 2 on your opponent's 2, compelling them to draw 4 cards. Should they augment the stack with another 2, you are then required to draw 6, and should the sequence continue, the maximum reaches 8.

4. Toss out a card with the number 7, and you can tell your opponent what the next card type should be. They gotta follow it. If they don't have a matching card, they pick up from the deck. Both players play by the same rule.

5. The winner is the one who dumps all their cards first.

Class diagram



Major Functions Overview

In this section, we will look at the significant functions in the game.

Each important function has a role to play in achieving the project objectives.

The following pages highlight the functionality, with sections of code and screenshots that provide a complete overview of how to implement it.

Major Functions Overview

lets start by the first function getting called to start a new game

```
void controller::startTheParty(){
//   game starting.......

    // reseting any previous values to start new game
    startNewGame();

    // putting first card to table
    putfirstcardtotable();

    // giving players first cards in the beginning of the game
    destributecardstoplayers();
    return;
}
```

This function as seen , uses 3 minor functions, that each have a single responsibility, such as start new game and distributing cards to players and table.

I tried my best to follow most best practices while coding , one of them is “The Single Responsibility Principle”

Major Functions Overview

First one takes care of initialising Deck, Players and Table.

```
void controller::startNewGame(){

    initialiseDeck();
    initialiseTable();
    initialisePlayers();

    return;
}
```

This function as seen , uses another 3 minor functions, that each have a single responsibility, such initializing Deck , Table and Players Instances.

Major Functions Overview

A deeper look into these Functions

```
void controller::initialiseTable(){
    tableCards->reset();
    return;
}

void controller::initialiseDeck(){
    // start new deck
    Deck->reset();
    Deck->resetNumberOfCardstoLift();
    Deck->shuffle();
    return;
}

void controller::initialisePlayers(){

    // giving the user the first turn to play
    p1->resetUSERHand();
    p1->setMyturn();
    p1->setfirstdist();

    // setting p2 as computer
    p2->resetCOMPUTERHand();
    p2->UnsetMyturn();

    return;
}
```

They basically use class-defined methods of those instances.
To control those instances, this is why I called it Controller Class, which is more like an MVC, where i separete the Qml view from the Logic.

Major Functions Overview

User Want to Put Card to Table

```
void controller::userWantToputcarttoTable(int num, QString name){
    //  here i follow the rules for the game , each card should be matching before puting any

    //  if lastcard == requested type thats fine , else { next card should match the requested
    if(p1->isItMyTurn()){

    QString lastcardinTable_Name = tableCards->getLastCardName();
    int lastCardinTbale_Number = tableCards->getLastCardNumber();
    QString Req = tableCards->getRequestedCardType();

    int putCardNumber=0;
    bool allowedCard = false;

    // before every thing , it should check if he last card was 2 , then check hand if it cont
    // i check if its matching the requested type , when he put 7 and request another type
    // if requested matching the last card added , mean he didnt put a 7 to request type chang

    if( Deck->getNumberOfCardsToLift() > 1 && num != 2 && lastCardinTbale_Number == 2){ ...

    if ( lastcardinTable_Name == Req ){

        if (lastcardinTable_Name == name || lastCardinTbale_Number == num){ ...}

        // if requested not matching the last card added , mean last player did put a 7 and re
    }else if(lastcardinTable_Name != Req){

        // only allowing cards that matches the request type
        if ( name == Req || num == 7){ ...
        }

        // tried to put else card than allowed .
        else{
//          # user tried to put a card differant than requested
        }
    }

    // if indeed he had a valid card to put down
    if (allowedCard && (putCardNumber != 0) ){

        if(p1->isMyHandEmpty()){
            setPlusxp(50); // when user wins ,he get 50xp
            emit gamesinishedChanged();// user won , end game
            return;
        }else if (!p1->isMyHandEmpty()){ ...}
        }

    return;
    }

    else{ ...}
}
```

As the name indicate , it get executed when user click to put card to table , so it should take care of controlling user clicks to only allow cards that are respect the game rules.

Major Functions Overview

User want to lift card from deck

```
void controller::userWanttoliftcardfromdeck(){

    if(p1->isItMyTurn()){
        liftcardfromdeck(*p1);// takes care for switching turn
        computerplay();
    }

    return;
}
```

as the name of the function indicates, it simply give a card to player p1 which is the instance of the player class.

Then it calls the Computer to play just after .

Major Functions Overview

Computer Play

as the name of the function indicates, it is the function made specifically for the computer turn, it’s a complicated function with a lot of nested if statements that control every scenario during the game , and depending on difficulty level set by user , it decide for the next move , while respecting the game rules.

The set of decisions that it can make based on difficulty level will be in details in the next page.

Major Functions Overview

Card 7

```
void controller::decideforseven(){
    QString y;
    if(getDifficulty() == "EASY"){
        // find card type that computer doesnt have , and 7kem biha
        y = p2->whatcardtypeidonthave();
    }
    else if(getDifficulty() == "MEDIUM"){
        // random , there is chance to have the randomly chosen type , same as there is a
        int x = QRandomGenerator::global()->bounded(4);
        switch (x) {
        case 0:
            y = "S";
            break;
        case 1:
            y = "J";
            break;
        case 2:
            y = "Z";
            break;
        case 3:
            y = "F";
            break;

        default:
            y = "F";
            break;
        }
    }
    else if(getDifficulty() == "HARD"){

        // dificulty hard , i will check if i have a 2 , i will request it type to put it
        if(p2->doihavedos()){
            // yes i have a two and i will request it type
            y = p2->cardtypeofdos();// find card index of a 2 card

        }else if ( ! p2->isMyHandEmpty()){

            // i dont have a 2 but i will request type of first card in my hand
            y = p2->typeoffirstcardinhand();

        }else if (p2->isMyHandEmpty()){

            // i dont have to request , since 7 was my last card ,i directly win
            emit gamesinishedChanged();
            return;
        }
    }
    // set requested type by computer
    tableCards->setRequestedCardType(y); // it takes care to signal qml
    return;
}
```

Decide for seven , the computer Decide for what type to request when he puts a card with number 7 , his decision based on difficulty level.

- Medium is a random choice.
- Easy requests a type that computer does not even have.
- Hard requests the most type it has in hand.

Major Functions Overview

Cards 1 and 2

similarly with card seven , computer decide for cards with number 1 and 2 , because they are special cards too , and a decision has to be made for them based on difficulty , such as computer wouldn’t be bothered to lift two cards when you put card number 2 in difficulty “easy”, in contrary for the difficulty “medium and hard ” where computer start looking in it hand if it got a card with number 2 to put on top of user’s 2 , to compel him to lift a 4 , 6 or an 8 depending on number of successive twos been put on top of each others.

For the card number 1 , it does not need a decision but it re-execute the computer play function so it is a case where the function get into a recursion state.

Major Functions Overview

Deck Constructor

Let’s move on from Controller and take a look on filldeck that get called when deck get constructed.

```
void deck::filldeck(){
    QString N = "SFJZ";
    for (int x = 0; x < 4; ++x) {

        QString S = N.mid(x, 1);
        //This code creates a new string S wi

        //
        for (int i = 1; i <= 7; ++i) {
            card c(i,S);
            deckOfCards.append(c);
        }

        //
        for (int i = 10; i <= 12; ++i) {
            card c(i,S);
            deckOfCards.append(c);
        }
        //
    }
    return;
}
```

The Magic hides behind the simplicity, as you can see that we have two loops, starting from 0 => 7 and from 10 => 12 so i skip creating 8th and 9th cards, that are not included in the Moroccan cards. it simply uses the card constructor to create instances on cards , and appends them to the QList of the Deck “deckOfCards”

Major Functions Overview

In Player Class

Let’s take a look on Put Down card and find Card index method , where i used method overloading technique.

```
// player will return only 1 card to the table , no more , s
card PutDownCArd(int num ,bool); // if it takes int and bool

card PutDownCArd(int const index);// if it take int only , m

card PutDownCArd(QString const name);// put card by matching
// put card matching either name or number, case doesnt matt
card PutDownCArd(int num , QString const name);

int findCardIndex(int const num ,QString const name)const;

int findCardIndex(int const num )const;

int findCardIndex(QString const name)const;
```

PutDownCard if a function serve same pupose but with different parametre types , so we can concentrate on putting the card , without worrying about function parametre should be given in order for it to work .

if you give it an integer as paramtre mean that it will count it as index and returns the card of that index, but if you give it an integer with a boolean , means that you want to return the cards with that specific card number , also when giving it a QString name ,it only return a card matching that name , in contrary whenyou give it name and number , so it has to return the exact card with that name and number .

Major Functions Overview

Table - Deck

If you are fimilliare with this game , you may already know that it may take a long time to finish the game , so we may need to lift more cards from deck than it size .

```
QList<card> & TableCards::giveBackMostCards(){
    emit shufflesignalChanged();

    tmpQ.clear();
        // i give back most cards
        int sz = tablecardss.size();

        for (int i= 0 ; i < sz-2 ; ++i){
        card tm(tablecardss[i]);
        tmpQ.append(tm);
     }

        card lst(tablecardss.last());
        tablecardss.clear();
        tablecardss.append(lst);

        return tmpQ;
}
```

As we used to take cards from table and leave only the last one on it , shuffle them again and return them to deck . i Made this function specifically for this Case , when it get called , it return most cards from table back to deck , so the game keep going non stop.

Relation Between C++ Functions and QML

We have been going through c++ methods without discussing about how they work in relation with QML in the front end .

```
Q_INVOKABLE void computerplayagain();

// slot to start the game from qml button
Q_INVOKABLE void startTheParty();

Q_INVOKABLE void userWantToputcarttoTable(int num , QString name);

Q_INVOKABLE void userWanttoliftcardfromdeck();

Q_INVOKABLE void userchosenewcardtype();

Q_INVOKABLE void changeCardStyle();

Q_INVOKABLE QString getcardStyle()const;

Q_INVOKABLE void saveScore();

Q_INVOKABLE void setDifficulty(const QString &newDifficulty);

Q_INVOKABLE int getPlusxp() const;

Q_INVOKABLE QString getUserName();

Q_INVOKABLE void setNewUserName(QString userName);
```

Did you notice how all these methods are Q_INVOKABLE.

Meaning that they can get called from the front end, so on click from the user can execute a function in QML , that uses one or more from those exposed functions to QML in order for the QML and C++ work together .

Relation Between C++ Functions and QML

But what if we need to update the view without having to wait a user interaction.

```
signals:

    void gamesinishedChanged();
    void DeckChanged();
    void p1Changed();
    void p2Changed();
    void tableChanged();
    void DifficultyChanged();
    void plusxpChanged();
    void sleepinqml();
};
```

Here Comes the importance of Signals, this feature what make QT a powerful C++ Framework.

Calling a Function that emits a signal to QML , to Update the View is an important feature ,it is more the reverse of Q_INVOKABLE.

So by combining those two key features it ensure a seamless collaboration between the frontend and the backend.

Mistakes Made
&
Lessons Learned

Since I Invested a lot of Time in This Project . I Did learn a lot of Things Such as :

- C++ Good Practices .
- QML Connections to C++ signals .
- Never give C++ pointer to QML .
- QT Framework Functionalities such as slots and signals and Q_Property and Q_INVOKABLE .

All those Combined Made My Software Development Skills to be Sharpened and Improved to a New Level.

I would like to express my gratitude to Professor **Ikram Ben Abdel Ouahab** for their guidance and teaching during this project. Their support has been invaluable to my learning and success

References

Qt 6 Documentation
https://doc.qt.io/qt-6/reference-overview.html

Qt 6 QML book
https://www.qt.io/product/qt6/qml-book

Qt Group
https://www.youtube.com/watch?v=rsBQgR_pd48

MontyTheSoftwareEngineer
https://www.youtube.com/watch?v=ragZPvRe6Pk

pouyaazimi
https://www.youtube.com/watch?v=Nma3c3YxsUo

KDAB
https://www.youtube.com/watch?v=JxyTkXLbcV4

VoidRealms
https://www.youtube.com/watch?v=nscgFv4l53w

ScytheStudio
https://www.youtube.com/watch?v=RWd-zjclFB0