Documentation

What is it?

This is a slot machine made based on a popular game in my country. It is a 3x5 reels slot machine that works almost the same as the real game. It has 5 paying lines and the RTP(Return to Player) of 95%. The game was made using a C++ back end and a QT front end. All the computations will be presented in a excel document that features number of winning combinations, their probability, the RTP given by each combination and the total RTP, as well as the symbols table.

Why did you choose this project?

I chose to make such a project because I am passionate about the mathematical aspects that it requires and I wanted to dive in deeper into the world of slots as I found it fascinating how the ‘random’ seems to be somehow expected. On top of that, I wanted to work with a front end language, this being my first project using an interface.

Details

On the main screen of the project, the reels can be seen as as well as the balance of the player, inputed from the code, the last win and the bet amount.

On top of that, there is a button that leads to the information screen where more details are given, info about the slot machine and on the paytable for the symbols. When getting a winning screen, the user can opt for the double or nothing feature that simulates the 50-50 using red/black cards. This page also has the history of the last 5 cards that were generated.

Main.qml

The file contains all the information about the front end of the main page. It has the typical qml buttons and the rectangles that have the cells for each reel. In order to simulate the spins, and the fast animation, a back end function that cycles through the images of the symbols was used. On top of that, 5 timers are used in order to get the sequential stop on each of the reels. Even though the symbols might seem random, they are not. They are randomly generated from a fixed symbols array: 3 adjacent symbols(numbers) are generated and shown on the screen. A matrix stores all the arrays of symbols and then the **randomfind** function generates 3 adjacent symbols for each row, which is then shown at the end of the animation. A second array is used to store the 3,4, or 5 winnings. So, for each line, there can be 3,4,5 matching symbols wins, and on top of that the 3,4,5 scatter wins, thus a 18 elements array is used to check which value from the paytable to be given to the player.

The line checks are done with the **getwins** function that simply checks for the corresponding cells if they are equal or not.

The winning amount is then calculated in the **calcwins** function that uses the getwins function. The nrforswitch gets the symbol that is in the corresponding cells of the screen and then it is passed on a if else-switch branch to calculate the total sum won.

Doublepage.qml

Represents the qml for the double or nothing page that has the black-red cards. Based on the winning amount, the user gets the option to double or nothing and when he goes on the page, a number between 0 and 1 is generated. Each red-black square on the page represents a number and in order to win, the user must press the block that matches the generated number. It is as simple as it is. The harder part about it was to get the animation for the flipping card, the computations to match what the user sees, aswell as the apparition of new cards on the history of the cards. In order to do that, a combination of listeners, timers is used to get the desired result.

Infopage.qml

The page that features information about the slot-it was the easiest to make as it did not require dynamic information, only to show on the screen the paytable and a bit of information about the slot, such as what are the paying lines, the RTP, and so on..

Conclusions

To conclude all, this was a swift go-through the project. I tried to explain the harder parts of it, that might not be very clear in the code as I myself find it hard to read after some time passed from finishing it. Even though the coding part was a bit hard for me given it was the first time experimenting with front end, I would say the hardest part about this was finding the mathematical model for a sot machine. There is not a lot of information about slot machines and how the RTP is calculated, nor in what way the random symbols are chosen, not even if the 50-50 is really a 50-50, given I saw that the number might be slightly on the edge for the house.

Guide to set-up the Slot Machine to be launchable on Android

(QT VERSION 6.9.1)

Must have the Android Kit which can be downloaded via the QT Maintenance Tool

Also, under there, or over Other Libraries, QTMultimedia must also be installed

For testing, Android Studio can be set-up for a device of type 86x64

Also, most likely the physical device, in case of android at least, would be arm64-v8a

After downloading Android for QT, the QTCreator app will prompt that there needs to be done a setup with the java JDK, android SDK, SSL settings, NDK components in order for developing to work.

Ill provide a link which helped me with the initial setup

<https://www.harshmittal.com/tutorials/Qt-Android-Deployment/>

IMPORTANT:

Choose a valid NDK for the current QT version or else, it will not automatically detect the allowed kits for the application. (for QT 6.9.1 I used NDK 27.3) (Java JDK version: 17).

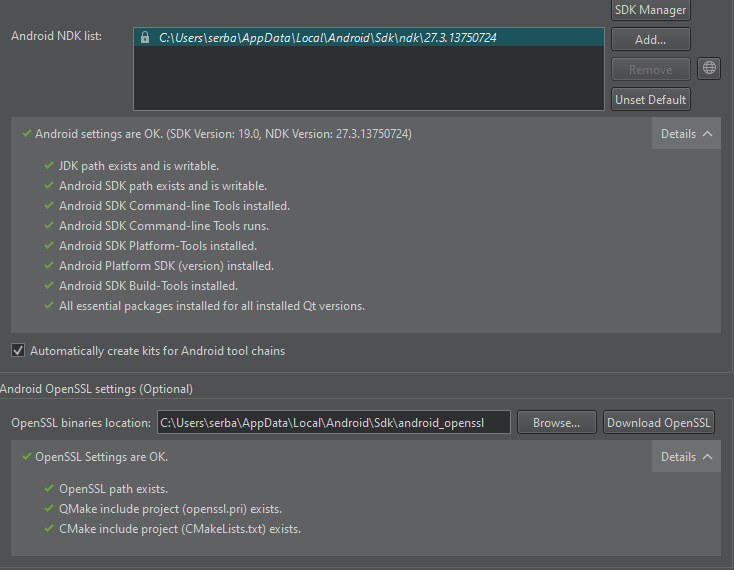
For the virtual android environment, I used a phone with SDK 34 (Android 14). Inside QT, SDK’s were for version 36

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

In order for it to work, it should look like this:



And under Kits-> Kits, it should look like this: A screenshot of a computer program

AI-generated content may be incorrect.

Nothing should be red inside KITS -> QTVersions

In order to test on the emulator, make sure it is open before, so it can be detected by qt

Build options can be found under Projects or when choosing what to build:

A screenshot of a computer program

AI-generated content may be incorrect.

On the physical device, there are a few prerequisites:

-enable developer options inside the settings, and under developer settings enable USB Debugging

-download android usb drivers, just search Samsung Android Usb Drivers (for the Samsung case)

-when connecting the phone via an usb cable, be sure to enable debugging on it when the pop-up appears on the screen

A screenshot of a computer

AI-generated content may be incorrect.-select your phone and hit build then run

->if you wanna share the app, get its apk (search it) and just tell other users to allow downloads from unknown sources