cmake\_minimum\_required(VERSION 3.16)

project(DrawWave VERSION 0.1 LANGUAGES CXX)

set(CMAKE\_AUTOMOC ON)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

find\_package(Qt6 6.2 COMPONENTS Quick REQUIRED)

qt\_add\_executable(appDrawWave

main.cpp

fileio.h

fileio.cpp

)

qt\_add\_qml\_module(appDrawWave

URI DrawWave

VERSION 1.0

QML\_FILES main.qml

)

set\_target\_properties(appDrawWave PROPERTIES

MACOSX\_BUNDLE\_GUI\_IDENTIFIER my.example.com

MACOSX\_BUNDLE\_BUNDLE\_VERSION ${PROJECT\_VERSION}

MACOSX\_BUNDLE\_SHORT\_VERSION\_STRING ${PROJECT\_VERSION\_MAJOR}.${PROJECT\_VERSION\_MINOR}

MACOSX\_BUNDLE TRUE

WIN32\_EXECUTABLE TRUE

)

target\_compile\_definitions(appDrawWave

PRIVATE $<$<OR:$<CONFIG:Debug>,$<CONFIG:RelWithDebInfo>>:QT\_QML\_DEBUG>)

target\_link\_libraries(appDrawWave

PRIVATE Qt6::Quick)

#ifndef FILEIO\_H

#define **FILEIO\_H**

#include <QObject>

class **FileIO** : public QObject

{

Q\_OBJECT

public:

Q\_PROPERTY(QString source

READ source

WRITE setSource

NOTIFY sourceChanged)

explicit **FileIO**(QObject \***parent** = 0);

Q\_INVOKABLE QString **read**();

Q\_INVOKABLE bool **write**(const QString& **data**);

QString **source**() { return mSource; };

public slots:

void **setSource**(const QString& **source**) { mSource = source; };

signals:

void **sourceChanged**(const QString& **source**);

void **error**(const QString& **msg**);

private:

QString **mSource**;

};

#endif // FILEIO\_H

#include "fileio.h"

#include <QFile>

#include <QTextStream>

FileIO::**FileIO**(QObject \***parent**) :

QObject(*parent*)

{

}

QString FileIO::**read**()

{

if (mSource.isEmpty()){

emit error("source is empty");

return QString();

}

QFile **file**(mSource);

QString **fileContent**;

if ( file.*open*(QIODevice::ReadOnly) ) {

QString **line**;

QTextStream **t**( &*file* );

do {

line = t.readLine();

fileContent += line;

} while (!line.isNull());

file.*close*();

} else {

emit error("Unable to open the file");

return QString();

}

return fileContent;

}

bool FileIO::**write**(const QString& **data**)

{

if (mSource.isEmpty())

return false;

QFile **file**(mSource);

if (!file.*open*(QFile::WriteOnly | QFile::Truncate))

return false;

QTextStream **out**(&*file*);

out << data;

file.*close*();

return true;

}

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include "fileio.h"

int **main**(int **argc**, char \***argv**[])

{

QGuiApplication **app**(*argc*, *argv*);

qmlRegisterType<FileIO, 1>("FileIO", 1, 0, "FileIO");

QQmlApplicationEngine **engine**;

const QUrl **url**(u"qrc:/DrawWave/main.qml"\_qs);

QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,

&app, [url](QObject \***obj**, const QUrl &**objUrl**) {

if (!obj && url == objUrl)

QCoreApplication::exit(-1);

}, Qt::QueuedConnection);

engine.load(url);

return app.exec();

}

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include "fileio.h"

int **main**(int **argc**, char \***argv**[])

{

QGuiApplication **app**(*argc*, *argv*);

qmlRegisterType<FileIO, 1>("FileIO", 1, 0, "FileIO");

QQmlApplicationEngine **engine**;

const QUrl **url**(u"qrc:/DrawWave/main.qml"\_qs);

QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,

&app, [url](QObject \***obj**, const QUrl &**objUrl**) {

if (!obj && url == objUrl)

QCoreApplication::exit(-1);

}, Qt::QueuedConnection);

engine.load(url);

return app.exec();

}

import QtQuick 2.15

import QtQuick.Window 2.15

import QtQuick.Controls 2.15

import QtCharts 6.2

import FileIO 1.0

Window {

width: 740

height: 480

visible: true

title: *qsTr*("Arnold's Wave Drawer")

property int timer: 0

FileIO {

id: *myFile*

source: "心电演示数据.csv"

}

Button {

id: *loadButton*

x: 627

y: 430

text: *qsTr*("加载数据")

onClicked: {

for (var *sig* in *myFile*.read())

{

*waveTextArea*.append(*sig*)

}

}

}

Button {

id: *saveButton*

x: 627

y: 377

text: *qsTr*("存储数据")

onClicked: {

*myFile*.write(*waveTextArea*.getText(0, 2047))

}

}

RadioButton {

id: *radioButton*

x: 627

y: 71

text: "静态显示"

checkable: false

}

RadioButton {

id: *radioButton1*

x: 627

y: 100

text: *qsTr*("动态显示")

checked: true

}

TextArea {

id: *waveTextArea*

x: 22

y: 305

width: 591

height: 165

placeholderText: *qsTr*("Wave Data")

background: Rectangle{

color: "gray"

}

}

ChartView {

id: *waveChartView*

x: 22

y: 36

width: 591

height: 264

antialiasing: true

backgroundColor: "#9917719b"

animationOptions: ChartView.SeriesAnimations

legend.visible:false

ValueAxis {

id: *myAxisX*

min: 0

max: 100

tickCount: 11

labelsColor: "#ffffff"

labelsFont.pointSize: 13

labelsFont.bold: true

labelFormat: '%d'

}

ValueAxis{

id:*myAxisY*

min:0

max:50

tickCount: 6

labelsColor: "#ffffff"

labelsFont.pointSize: 13

labelsFont.bold: true

labelFormat: '%d'

}

LineSeries {

id:*lineSeries*

name: "LineSeries"

axisX: *myAxisX*

axisY:*myAxisY*

color: "#00ffff"

width: 3

}

}

Timer{

interval: 10

running: true

repeat: true

onTriggered: {

*lineSeries*.append(*timer*,*Math*.random()\*50)

if (*timer* > 100)

{

*timer* = 0

*lineSeries*.clear()

}

else

{

*timer* += 1

}

}

}

}

