# 图形绘画小程序

## 19084129-李奕澄

Widget.h

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <QtCharts>

using namespace QtCharts;

namespace Ui {

class Widget;

}

class Widget : public QWidget

{

Q\_OBJECT

public:

explicit Widget(QWidget \*parent = nullptr);

~Widget();

private slots:

void on\_readDataButton\_clicked();

void on\_saveDataButton\_clicked();

void on\_staticRadioButton\_clicked();

void on\_dynamicRadioButton\_clicked();

void timeOutAction();

private:

Ui::Widget \*ui;

QStringList mCSVList;

QValueAxis \*mAxisY;

QValueAxis \*mAxisX;

QLineSeries \*mSeries;

QChart \*mWaveLineChart;

QTimer \*mDrawWaveTimer;

int mOriginListIndex;

bool mFirstDraw;

QVector<QPointF> mWavePointBuffer;

void initWaveLineChart();

void drawWave(int axisX,qint16 data);

};

#endif

Widget.cpp

#include "widget.h"

#include "ui\_widget.h"

#include <QtCharts/QChartView>

#include <QtCharts/QLineSeries>

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

setWindowTitle(QString::fromUtf8("窗口画图小程序"));

ui->procDataProgressBar->setVisible(false);

initWaveLineChart();

mOriginListIndex=0;

mFirstDraw=true;

mDrawWaveTimer=new QTimer(this);

connect(mDrawWaveTimer,SIGNAL(timeout()),this,SLOT(timeOutAction()));

}

Widget::~Widget()

{

delete ui;

}

void Widget::on\_readDataButton\_clicked()

{

QString readDataString;

int i=0;

QString fileName=QFileDialog::getOpenFileName(this,tr("Open File"),"","CSV Files(\*.csv)",0);

if(!fileName.isNull())

{//fileName为文件名

qDebug() << fileName;

QFile csvFile(fileName);

mCSVList.clear();

if(csvFile.open(QIODevice::ReadWrite))

{

QTextStream stream(&csvFile);

while (!stream. atEnd())

{

mCSVList.push\_back(stream.readLine());

}

csvFile.close();

}

ui->procDataProgressBar->setRange(0, mCSVList.length()-1);

ui->procDataProgressBar->setValue(0);

ui->procDataProgressBar->setVisible(true);

Q\_FOREACH(QString str,mCSVList)

{

ui->procDataProgressBar->setValue(1);

i++;

readDataString += str + " ";

}

ui->procDataProgressBar->setVisible(false);

ui->dataPlainTextEdit->clear();

ui->dataPlainTextEdit->appendPlainText(readDataString);

mWavePointBuffer.clear();

ui->staticRadioButton->setChecked(true);

if(mCSVList.length() >= 2048)

{

for(int j = 0; j < 2048; j++)

{

mWavePointBuffer.append(QPointF(j,mCSVList.at(j).toInt()));

}

}

else

{

for(int k = 0; k < mCSVList. length(); k++)

{

mWavePointBuffer.append(QPointF(k,mCSVList.at(k).toInt()));

}

}

mSeries->replace(mWavePointBuffer);

}

else {

//选择取消

qDebug() << "cancel";

}

}

void Widget::on\_saveDataButton\_clicked()

{

QString fileName =QFileDialog::getSaveFileName(this,tr("save file"),"",tr("CSV Files (\*.csv)"));

QStringList saveString;

int i;

if(!fileName.isNull())

{

//fileName 为文件名

qDebug() << fileName;

QFile file(fileName);//文件命名

saveString = ui->dataPlainTextEdit->toPlainText().split(" ");

if(file.open(QFile::WriteOnly | QFile::Text))

{//检查文件是否打开

QTextStream out(&file);

ui->procDataProgressBar->setRange(0, saveString.length() - 1);

ui->procDataProgressBar->setValue(0);

ui->procDataProgressBar->setVisible(true);

Q\_FOREACH(QString str, saveString)

{

out << str + "\n";

ui->procDataProgressBar->setValue(i);

i++;

}

ui->procDataProgressBar->setVisible(false);

file. close();

}

}

else

{

qDebug() << "cancel";

}

}

void Widget::on\_staticRadioButton\_clicked()

{

qDebug()<<"staticRadioButton click";

if(mDrawWaveTimer->isActive())

{

mDrawWaveTimer->stop();

mWavePointBuffer.clear();

if (mCSVList.length() >= 2048)

{

for(int i = 0; i < 2048; i++)

{

mWavePointBuffer.append(QPointF(i, mCSVList.at(i).toInt()));

}

}

else

{

for(int i = 0;i < mCSVList.length(); i++)

{

mWavePointBuffer.append(QPointF(i, mCSVList.at(i).toInt()));

}

}

mSeries->replace(mWavePointBuffer);

}

}

void Widget::on\_dynamicRadioButton\_clicked()

{

qDebug() << "dynamicRadioButton click";

if (mCSVList.length() == 0)

{

qDebug() << "null";

return;

}

if(!mDrawWaveTimer->isActive())

{

mOriginListIndex = 0;

mFirstDraw =true;

mDrawWaveTimer->start(1);//1ms执行一次

mWavePointBuffer.clear();

}

}

void Widget::timeOutAction()

{

if(mOriginListIndex > (mCSVList.length() -1))

{

mOriginListIndex=0;

mFirstDraw = false;

}

qint16 tempInt16=mCSVList.at(mOriginListIndex).toInt();

drawWave(mOriginListIndex,tempInt16);

mOriginListIndex++;

}

void Widget::initWaveLineChart()

{

mAxisX = new QValueAxis();

mAxisY = new QValueAxis();

mSeries = new QLineSeries();

mWaveLineChart = new QChart();

mWaveLineChart->addSeries(mSeries);

mAxisX->setRange(0,2048);

mAxisY->setRange(0,4096);

mAxisX->setGridLineVisible(false);

mAxisX->setLabelsVisible(false);

mAxisY->setLabelsVisible(false);

mWaveLineChart->addAxis(mAxisX, Qt::AlignBottom) ;

mWaveLineChart->addAxis(mAxisY,Qt::AlignLeft);

mSeries->attachAxis (mAxisX);

mSeries->attachAxis(mAxisY);

mSeries->setColor(QColor(Qt::black));

mWaveLineChart->layout()->setContentsMargins(0,0,0,0);

mWaveLineChart->setMargins(QMargins(0,0,0,0));

mWaveLineChart->setBackgroundRoundness(0);

mWaveLineChart->legend()->hide();

ui->waveGraphicsView->setChart(mWaveLineChart);

}

void Widget::drawWave(int axisX,qint16 data)

{

int timesCounts = axisX / 2048; //计算 数据是否超过了2048

if(timesCounts > 0 || mFirstDraw == false)

{

axisX = axisX - timesCounts \* 2048;

mWavePointBuffer[axisX].setY(data);

}

else//如果是第一次界面

{

mWavePointBuffer.append(QPointF (axisX,data));

}

mSeries->replace (mWavePointBuffer);

}

实验结果：

