#pragma once

#include <stdint.h>

#include <ESEDecode\ESEFile.h>

using namespace std;

namespace OESE

{

#define MAX\_ESE\_WAV\_NUMS 9

#define ESE\_PTS 128

#define MAX\_SPEED 256

#define MAX\_RPM 8192

#define MAX\_PEDAL 128

#define MAX\_TORQUE 512

/\*---------------车外声浪-------------------\*/

//车外声浪文件开始 24 bytes

struct OeseFileHeader

{

//Acoutec\_OESE\_Version\_1\_0 OESE 车外声浪

char head[24] = { 0x41,0x63,0x6F,0x75,0x74,0x65,0x63,0x5F,0x4F,0x45,0x53,0x45,0x5F,0x56,0x65,0x72,0x73,0x69,0x6F,0x6E,0x5F,0x31,0x5F,0x30 };

};

//车外声浪文件结尾 20 bytes

struct OeseFileFooter

{

//Copyright\_by\_Acoutec

char tag[20] = {0x43,0x6F,0x70,0x79,0x72,0x69,0x67,0x68,0x74,0x5F,0x62,0x79,0x5F,0x41,0x63,0x6F,0x75,0x74,0x65,0x63 };

};

//wav参数文件导入结构体 4376 bytes

struct OeseShiftPara

{

unsigned int mute\_t;//4byte

float gain\_f;//4byte

unsigned int pitchDep\_i;//4byte=1

float pitchMapper\_l[ESE\_PTS];//4\*128byte

unsigned int gain1dep\_i;//4byte

float gain1mapper\_l[ESE\_PTS];//4\*128byte

unsigned int gain2dep\_i;//4byte

float gain2mapper\_l[ESE\_PTS];//4\*128byte

};

struct OESEFile

{

void init()

{

memset(this, 0, sizeof(OESEFile));

}

OeseFileHeader header;//文件头 24 bytes

unsigned int lens[MAX\_ESE\_WAV\_NUMS];//15个文件的音频数据长度 60 bytes 2个类阶次+9个普通wav 长度为0则不使用

OeseShiftPara importPara[MAX\_ESE\_WAV\_NUMS];//9个外部导入的wav参数 1556\*9 = 14004 bytes

Eq eq;

ESE orderFile;//阶次文件结构体(box) 12492 bytes

char name[MAX\_ESE\_WAV\_NUMS][128];

char\* data[MAX\_ESE\_WAV\_NUMS];//9个音频文件的数据 94kb\*9 = 846 kb

OeseFileFooter footer;//文件尾 20 bytes

};

//oese 参数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*mute :0 represent mute , 1 represent not mute\*/

/\*gain :9 wav all gain \*/

/\*pitchDep :9 wav shift type \*/

/\*gain0Dep :9 wav first gain type \*/

/\*gain1Dep :9 wav second gain type \*/

/\*pitch : 9 wav shift curve \*/

/\*gain0 : 9 wav first gain curve \*/

/\*gain1 : 9 wav second gain curve \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Associated Speed Signal Type \*/

/\*0->None :none associated \*/

/\*1->Speed :0-256km/h one point represent 2km/h\*/

/\*2->RPM :0-8192 one point represent 64 \*/

/\*3->Pedal :0-128 one point represent 1 \*/

/\*4->Torque :0-512 one point represent 4N\*M \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct SAF\_PARA

{

uint32\_t mute[MAX\_ESE\_WAV\_NUMS];

float gain[MAX\_ESE\_WAV\_NUMS];//WAV gain

uint32\_t pitchDep[MAX\_ESE\_WAV\_NUMS];//0:None;1:Speed;2:RPM;3:Pedal;4:Torque

uint32\_t gain0Dep[MAX\_ESE\_WAV\_NUMS];

uint32\_t gain1Dep[MAX\_ESE\_WAV\_NUMS];

float pitch\_x[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//pitch

float pitch\_y[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//pitch

float gain0\_x[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//gain0

float gain0\_y[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//gain0

float gain1\_x[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//gain1

float gain1\_y[MAX\_ESE\_WAV\_NUMS][ESE\_PTS/2];//gain1

};

}

#pragma execution\_character\_set("utf-8")

#include "OesePlayerView.h"

#include "Helper/QFileDialogHelper.h"

#include "Base/MesseageBox.h"

#include "AudioDecode\Encoders.h"

using namespace std;

using namespace audio;

OesePlayerView::OesePlayerView(QWidget \*parent)

: QWidget(parent)

, ui(new Ui::OesePlayerViewClass())

, m\_oesePlayer(new OesePlayer())

, m\_timer(new QTimer())

{

ui->setupUi(this);

initChart();

connectSlots();

}

OesePlayerView::~OesePlayerView()

{

delete ui;

delete m\_timer;

}

void OesePlayerView::set\_project(boost::shared\_ptr<ESEDesignProject> & project)

{

m\_project = project;

}

void OesePlayerView::connectSlots()

{

connect(ui->cbx\_play\_mode, static\_cast<void(QComboBox::\*)(int)>(&QComboBox::currentIndexChanged), this, &OesePlayerView::setPlayMode);

connect(ui->exportFlie, &QToolButton::clicked, this, &OesePlayerView::exportCurve);

connect(ui->importFile, &QToolButton::clicked, this, &OesePlayerView::importCurve);

connect(ui->btn\_play, &QToolButton::clicked, this, &OesePlayerView::playOrStop);

connect(ui->btn\_test, &QToolButton::clicked, this, &OesePlayerView::autoTest);

connect(m\_timer, &QTimer::timeout, this, &OesePlayerView::onTimeOut);

connect(ui->btn\_export\_wav, &QToolButton::clicked, this, &OesePlayerView::saveWav);

connect(ui->doubleSpinBox, static\_cast<void(QDoubleSpinBox::\*)(double)>(&QDoubleSpinBox::valueChanged), this, [=](double value) {

delay = value;

});

}

void OesePlayerView::initChart()

{

ui->vehicleSpeedWidget->setGraphType("车速km/h");

ui->vehicleRpmWidget->setGraphType("转速r/min");

ui->vehiclePedalWidget->setGraphType("踏板 °");

ui->vehicleTorqueWidget->setGraphType("扭矩 N");

ui->vehicleSpeedWidget->setYRange(0,MAX\_SPEED);

ui->vehicleRpmWidget->setYRange(0,MAX\_RPM);

ui->vehiclePedalWidget->setYRange(0,MAX\_PEDAL);

ui->vehicleTorqueWidget->setYRange(0,MAX\_TORQUE);

ui->vehicleSpeedWidget->setDoubleSpinBoxRange(MAX\_SPEED);

ui->vehicleRpmWidget->setDoubleSpinBoxRange(MAX\_RPM);

ui->vehiclePedalWidget->setDoubleSpinBoxRange(MAX\_PEDAL);

ui->vehicleTorqueWidget->setDoubleSpinBoxRange(MAX\_TORQUE);

}

void OesePlayerView::setPlayMode(int index)

{

m\_oesePlayer->set\_play\_mode(index);

}

void OesePlayerView::importCurve()

{

QString filename = QFileDialogHelper::getOpenFileName(this, "导入序列", "\*.txt");

if (filename.isEmpty())

return;

fstream file(filename.toLatin1().data(), ios::in);

if(!file.is\_open())

{

MesseageBox::showError("文件打开失败");

}

string buf = "";

int line = 0;

QVector<QPointF> points;

while (getline(file, buf))

{

if (buf == "speedData" || buf == "rpmData" || buf == "pedalData" || buf == "torqueData")

{

continue;

}

QStringList list = QString::fromStdString(buf).split(",");

points.push\_back(QPointF(list.at(0).toDouble(), list.at(1).toDouble()));

line++;

}

if (line <= 4000 - 1)

{

MesseageBox::showError("文件已损坏");

return;

}

speedData.clear();

rpmData.clear();

pedalData.clear();

torqueData.clear();

int i = 0;

for (auto point : points)

{

if (i < 1000)

speedData.push\_back(points.at(i));

else if (i >= 1000 && i < 2000)

rpmData.push\_back(points.at(i));

else if (i >= 2000 && i < 3000)

pedalData.push\_back(points.at(i));

else

torqueData.push\_back(points.at(i));

i++;

}

ui->vehicleSpeedWidget->setCurveData(speedData);

ui->vehicleRpmWidget->setCurveData(rpmData);

ui->vehiclePedalWidget->setCurveData(pedalData);

ui->vehicleTorqueWidget->setCurveData(torqueData);

MesseageBox::showInfo("导入成功");

}

void OesePlayerView::synchroCurve()

{

speedData = ui->vehicleSpeedWidget->getCurveData();

rpmData = ui->vehicleRpmWidget->getCurveData();

pedalData = ui->vehiclePedalWidget->getCurveData();

torqueData = ui->vehicleTorqueWidget->getCurveData();

}

void OesePlayerView::exportCurve()

{

QString filename = QFileDialogHelper::getSaveFileName(this, "导出序列", "\*.txt");

if (filename.isEmpty())

{

return;

}

fstream dataFile(filename.toLatin1().data(), ios::out | ios::trunc);

synchroCurve();

dataFile << "speedData" << endl;

for (QPointF point : speedData)

{

dataFile << point.x() << "," << point.y() << endl;

}

dataFile << "rpmData" << endl;

for (QPointF point : rpmData)

{

dataFile << point.x() << "," << point.y() << endl;

}

dataFile << "pedalData" << endl;

for (QPointF point : pedalData)

{

dataFile << point.x() << "," << point.y() << endl;

}

dataFile << "torqueData" << endl;

for (QPointF point : torqueData)

{

dataFile << point.x() << "," << point.y() << endl;

}

dataFile.close();

MesseageBox::showInfo("导出成功");

}

void OesePlayerView::playOrStop(bool clicked)

{

ui->btn\_test->setEnabled(clicked);

if (clicked)

{

m\_oesePlayer->setOeseFile(m\_project->get\_oese\_ptr());

m\_oesePlayer->set\_play\_mode(ui->cbx\_play\_mode->currentIndex());

m\_oesePlayer->start();

} else {

ui->btn\_test->setChecked(false);

m\_timer->stop();

m\_oesePlayer->stop();

ui->vehicleSpeedWidget->stop();

ui->vehicleRpmWidget->stop();

ui->vehiclePedalWidget->stop();

ui->vehicleTorqueWidget->stop();

}

}

void OesePlayerView::autoTest(bool clicked)

{

if (m\_timer->isActive()) {

m\_timer->stop();

}

if (clicked) {

synchroCurve();

m\_timer->start(delay); //ms

count = 0;

}

}

void OesePlayerView::onTimeOut()

{

if (count == 999)

{

ui->btn\_play->setChecked(false);

ui->btn\_test->setChecked(false);

playOrStop(false);

return;

}

ui->vehicleSpeedWidget->start(count);

ui->vehicleRpmWidget->start(count);

ui->vehiclePedalWidget->start(count);

ui->vehicleTorqueWidget->start(count);

m\_oesePlayer->setVehicle((uint16\_t)(speedData[count].y()));

m\_oesePlayer->setRpm((uint64\_t)(rpmData[count].y()));

m\_oesePlayer->setPedal((uint8\_t)(pedalData[count].y()));

m\_oesePlayer->setTorque((uint16\_t)(torqueData[count].y()));

count++;

}

void OesePlayerView::saveWav()

{

QVector<short> data;

m\_oesePlayer->get\_record\_data(data);

if (data.isEmpty())

return;

QString save\_path = QFileDialogHelper::getSaveFileName(this, "选择存储路径", "\*.wav");

if (save\_path.isEmpty())

return;

ma\_encoder\_config config = ma\_encoder\_config\_init(ma\_resource\_format\_wav, ma\_format\_s16, 1, 48000);

ma\_encoder encoder;

encoder.pUserData = (void \*)data.constData();

ma\_result result = ma\_encoder\_init\_file(save\_path.toLocal8Bit().toStdString().c\_str(), &config, &encoder);

if (result != MA\_SUCCESS) {

MesseageBox::showError("保存失败,Error code("+QString::number(result)+")");

}

int ret = ma\_encoder\_write\_pcm\_frames(&encoder, data.constData(), data.size());

ma\_encoder\_uninit(&encoder);

MesseageBox::showInfo("保存成功");

}

#pragma execution\_character\_set("utf-8")

#include "GainView.h"

GainView::GainView(QWidget \*parent)

: QDialog(parent)

{

ui = new Ui::GainView();

ui->setupUi(this);

initChart();

}

GainView::~GainView()

{

delete ui;

}

void GainView::closeEvent(QCloseEvent \*event)

{

double pitchCurve1[ESE\_PTS] = { 0 };

int pitchType = 0;

double gainCurve2[ESE\_PTS] = { 0 };

int gainType2 = 0;

double gainCurve3[ESE\_PTS] = { 0 };

int gainType3 = 0;

int allGain = ui->sb\_all\_gain->value();

ui->pitch\_widget->getData(pitchCurve1, pitchType);

ui->gain\_widget\_1->getData(gainCurve2, gainType2);

ui->gain\_widget\_2->getData(gainCurve3, gainType3);

emit dataReady(pitchCurve1, pitchType, gainCurve2, gainType2, gainCurve3, gainType3, allGain);

event->accept();

}

void GainView::initChart()

{

//设置频移界面参数

ui->pitch\_widget->setGraphTitle("PITCHSHIFT");

ui->pitch\_widget->setGraphCurveType(GRAPHTYPE::PITCH\_);

//设置增益界面参数

ui->gain\_widget\_1->setGraphTitle("GAIN");

ui->gain\_widget\_1->setGraphCurveType(GRAPHTYPE::GAIN\_);

ui->gain\_widget\_2->setGraphTitle("GAIN");

ui->gain\_widget\_2->setGraphCurveType(GRAPHTYPE::GAIN\_);

}

void GainView::dataUpdate(double \*pitchCurve1, int pitchType,

double \*gainCurve2, int gainType2, double \*gainCurve3, int gainType3, int allGain)

{

ui->pitch\_widget->setData(pitchCurve1, pitchType);

ui->gain\_widget\_1->setData(gainCurve2, gainType2);

ui->gain\_widget\_2->setData(gainCurve3, gainType3);

ui->sb\_all\_gain->setValue(allGain);

}

#pragma execution\_character\_set("utf-8")

#include "WavItem.h"

#include "Base/MesseageBox.h"

#include "Helper/QFileDialogHelper.h"

WavItem::WavItem(int num, QWidget \*parent)

: QWidget(parent)

, ui(new Ui::WavItemClass())

, gainView(new GainView())

{

ui->setupUi(this);

ui->pushButton->setText(QString::number(num));

connectSlots();

//gainView->hide();

}

WavItem::~WavItem()

{

delete ui;

}

void WavItem::connectSlots()

{

connect(ui->btn\_open, &QToolButton::clicked, this, &WavItem::openWave);

connect(ui->btn\_set, &QToolButton::clicked, this, &WavItem::setShiftPara);

connect(gainView.get(), &GainView::dataReady, this, &WavItem::updateShiftPara);

connect(ui->pushButton, &QPushButton::clicked, this, [=]() {

if (!ui->btn\_enable->isChecked())

{

MesseageBox::showInfo("请先使能本项");

return;

}

if (MesseageBox::showQuestion("是否确定删除？")) {

wavData.init();

ui->label\_name->clear();

}

});

connect(ui->btn\_enable, &QToolButton::clicked, [=](bool checked)

{

if (checked)

{

ui->widget->setStyleSheet("background-color: #FFFFFF;");

}

else

{

ui->widget->setStyleSheet("background-color: #C7C7C7;");

}

});

}

void WavItem::openWave()

{

if (!ui->btn\_enable->isChecked())

{

MesseageBox::showInfo("请先使能本项");

return;

}

QString file = QFileDialogHelper::getOpenFileName(this, "选择wav", "\*.wav|(\*.wav)");

if (!file.isEmpty())

{

filename = file;

QFileInfo info(file);

ui->label\_name->setText(info.fileName());

//加载wav

audio::AudioDecoder decoder;

std::string stdFile = file.toLocal8Bit().toStdString();

if (decoder.IsFileSupported(stdFile))

{

decoder.Load(&wavData, stdFile);

}

else

{

MesseageBox::showError("不支持的音频文件");

}

}

}

void WavItem::setShiftPara()

{

if (ui->btn\_enable->isChecked())

gainView->exec();

else

MesseageBox::showInfo("请先使能本项");

}

void WavItem::updateShiftPara(double \*pitchCurve1, int pitchType,

double \*gainCurve2, int gainType2, double \*gainCurve3, int gainType3, int allGain)

{

memcpy(this->pitchCurve1, pitchCurve1, sizeof(double)\*ESE\_PTS);

this->pitchType = pitchType;

memcpy(this->gainCurve2, gainCurve2, sizeof(double)\*ESE\_PTS);

this->gainType2 = gainType2;

memcpy(this->gainCurve3, gainCurve3, sizeof(double)\*ESE\_PTS);

this->gainType3 = gainType3;

this->allGain = allGain;

emit dataUpdate();

}

void WavItem::getData(std::vector<uint8\_t> &wav\_data, int &wav\_len, bool &mute,

double \*pitchCurve1, int &pitchType,

double \*gainCurve2, int &gainType2,

double \*gainCurve3, int &gainType3,

int &allGain, QString &name)

{

for (size\_t i = 0; i < wavData.data->size(); i++)

{

wav\_data.push\_back((\*(wavData.data))[i]);

}

wav\_len = wavData.dataLength;

mute = ui->btn\_enable->isChecked();

for (size\_t i = 0; i < ESE\_PTS; i++)

{

pitchCurve1[i] = this->pitchCurve1[i];

gainCurve2[i] = this->gainCurve2[i];

gainCurve3[i] = this->gainCurve3[i];

}

pitchType = this->pitchType;

gainType2 = this->gainType2;

gainType3 = this->gainType3;

allGain = this->allGain;

name = ui->label\_name->text();

}

void WavItem::setData(std::vector<uint8\_t> &wav\_data, int &wav\_len, bool &mute,

double \*pitchCurve1, int &pitchType,

double \*gainCurve2, int &gainType2,

double \*gainCurve3, int &gainType3,

int &allGain, QString &name)

{

for (size\_t i = 0; i < wav\_data.size(); i++)

{

wavData.data->push\_back(wav\_data[i]);

}

wavData.dataLength = wav\_len;

ui->btn\_enable->setChecked(mute);

if (mute)

{

ui->widget->setStyleSheet("background-color: #FFFFFF;");

}

else

{

ui->widget->setStyleSheet("background-color: #C7C7C7;");

}

for (size\_t i = 0; i < ESE\_PTS; i++)

{

this->pitchCurve1[i] = pitchCurve1[i];

this->gainCurve2[i] = gainCurve2[i];

this->gainCurve3[i] = gainCurve3[i];

}

this->pitchType = pitchType;

this->gainType2 = gainType2;

this->gainType3 = gainType3;

this->allGain = allGain;

this->filename = name;

if (wav\_data.size())

gainView->dataUpdate(this->pitchCurve1, this->pitchType,

this->gainCurve2, this->gainType2,

this->gainCurve3, this->gainType3,

this->allGain);

ui->label\_name->setText(name);

}

#pragma once

#include <QDialog>

#include "ui\_eqsetting.h"

#include <boost/signals2.hpp>

#include "OeseFileTypes.h"

using namespace OESE;

QT\_BEGIN\_NAMESPACE

namespace Ui { class EqSettingClass; };

QT\_END\_NAMESPACE

class EqSetting : public QDialog

{

Q\_OBJECT

public:

EqSetting(QWidget \*parent = nullptr);

~EqSetting();

void set\_data(Eq eq);

void get\_data(Eq &eq);

virtual void closeEvent(QCloseEvent\* event) override;

boost::signals2::signal<void()> eqClose;

private:

Ui::EqSettingClass \*ui;

QVector<QPair<QSlider\*, QDoubleSpinBox\*>> group;

float eq\_data[6][10] = { 0 };

bool isOpen = false;

int cur\_channel = 0;

};