高级语言程序设计

实验报告

南开大学 工科试验班

姓名：黄婉文

学号：2413406

班级：3-1班

2025年3月19日

目录

[高级语言程序设计大作业实验报告 1](#_Toc21286)

[一. 作业题目 1](#_Toc29969)

[二. 开发软件 1](#_Toc18364)

[三. 课题要求 1](#_Toc742)

[四. 主要流程 1](#_Toc25240)

[1． 实现流程 1](#_Toc20452)

[2． 核心逻辑与代码 2](#_Toc25354)

[（1） 开始界面 2](#_Toc22508)

[（2） 模式选择界面 2](#_Toc28665)

[（3） 游戏主界面 3](#_Toc32428)

[五. 单元测试与常见问题 6](#_Toc28265)

[六. 收获 6](#_Toc23116)

[1． 控件与信号槽 6](#_Toc25354)

[2. 音乐播放 6](#_Toc28665)

[3. 计时器 6](#_Toc28665)

[4. 初始化 7](#_Toc28665)

高级语言程序设计大作业实验报告

1. **作业题目**

用Qt制作简易古风节奏类音乐游戏。

1. **开发软件**

Qt Creator 5.14.2

Visual Studio 2022

1. **课题要求**
2. 面向对象。
3. 单元测试。
4. 模型部分
5. 验证
6. **主要流程**
   1. **整体流程**
7. 首先通过设计师界面设计主要的ui界面，包括开始页面、模式选择、游戏主界面。
8. 准备音乐、图片等素材。
9. 添加QPushButton、QLabel、graphicsview、radiobutton等基本控件并添加信号和槽，实现基本的页面跳转等功能。
10. 通过import Qtmultimedia插入Qmusicplayer插件，实现音乐播放。
11. 绘制背景、判定线、轨道分割线、音符，根据轨道确定音符下落坐标。
12. 定义note和notedata结构体，并使用QList存储。
13. 设置键盘事件，按下对应按键移除判定线附近的音符。
14. 绘制谱面。
15. 实现暂停与继续，用lcdnumber进行得分统计与显示。
16. 测试与优化。
    1. **核心逻辑与代码**
17. 开始界面：通过qpushbotton的槽连接到下一个页面。

void MainWindow::**on\_pushButton\_clicked**()

{

this->close();

m=new mode;

m->show();

}

1. 模式选择界面：通过radiobutton实现单选，并将各单选按钮的槽连接到对应谱面。同时弹出Qmessagebox，提示游戏玩法。

int flag=0;

void mode::**on\_radioButton\_toggled**(bool checked)

{

if(checked)

flag=1;

}

void mode::**on\_radioButton\_2\_toggled**(bool checked)

{ if(checked)

flag=2;

}

void mode::**on\_pushButton\_clicked**()

{

if(flag==1){

this->close();

g=new gamewindow;

g->show();

QMessageBox::information(this,"教程","键盘A,S,K,L分别控制从左到右的轨道");}

//打开谱面1

1. 游戏主界面：

设置四条轨道，并将每条轨道的中央坐标设为常变量，作为音符下落的位置。

const int a=320;

const int b=590;

const int c=860;

const int d=1130;

用Qtimer设置定时器用于计时，以播放音乐、定时更新游戏逻辑。

// 启动定时器

gameTimer = new QTimer(this);

connect(gameTimer, &QTimer::timeout, this, &gamewindow::updateGame);

声明musicplayer变量，设置好音量，播放音乐。

musicplayer=new QMediaPlayer;

musicplayer->setMedia(QUrl("qrc:/opera.mp3"));

musicplayer->setVolume(70);

在头文件中定义音符和音符数据结构体，包括音符大小、颜色、出现时间、坐标（轨道）等，并在源文件中初始化。

头文件：struct **Note** {

int x;

int y;

int width;

int height;

QColor color;

};

struct **NoteData** {

qint64 time; // 音符出现的时间（毫秒）

int x; // 音符的横坐标

bool generated; // 是否已生成

};

源文件：// 初始化音符数据

noteDataList.append({ });

编写游戏开始逻辑和更新逻辑，包括开始时间、重绘时间间隔、得分记录等。注意函数要先在头文件中声明。

void gamewindow::**startGame**() {

qDebug()<<"start";

gameTimer->start(16); // 16ms 间隔，约 60 FPS

updateGame();

// 游戏启动逻辑

}

void gamewindow::**updateGame**() {

updateGameLogic(); // 更新游戏逻辑

update(); // 触发重绘

ui->scorenumber->display(score);

}

void gamewindow::**updateGameLogic**() {

qint64 currentTime = musicplayer->position(); // 获取当前播放时间

// 遍历音符数据，生成音符

for (auto &noteData : noteDataList) {

if (noteData.time <= currentTime && !noteData.generated) {

notes.append({noteData.x, 0, 70, 20,QColor(110, 120, 112, 255)}); // 生成音符

noteData.generated = true; // 标记为已生成

}

}

// 更新音符位置

for (auto &note : notes) {

note.y += 10; // 让音符下落

}

}

用QPainter绘制音符和背景,用rgba设置渐变色。

void gamewindow::***paintEvent***(QPaintEvent \*event) {

QPainter painter(this);

// 只绘制音符

drawNotes(&painter);

}

void gamewindow::**drawNotes**(QPainter \*painter) {

// 创建线性渐变

QLinearGradient gradient(0, 0, 0, height());

gradient.setColorAt(0, Qt::white); // 起始颜色

gradient.setColorAt(1, QColor(149, 142, 108, 255)); // 结束颜色

// 使用渐变填充背景

painter->fillRect(rect(), gradient);

for (const auto &note : notes) {

QBrush noteBrush(note.color);

painter->setBrush(noteBrush);

QRect noteRect(note.x, note.y, note.width, note.height);

painter->drawRect(noteRect);

}

编写键盘事件，当音符在判定线附近时按下对应按键便移除音符，并积一分。

void gamewindow::***keyPressEvent***(QKeyEvent \*event) {

int key = event->key(); // 获取按下的按键

// 遍历音符，检查是否匹配按键

for (auto it = notes.begin(); it != notes.end(); ) {

if (key == Qt::Key\_A && (it->x == a&&it->y>=800&&it->y<=1100)) { // 按下 A 键对应 x=100 的音符

it = notes.erase(it); // 移除音符

score++;

} else if (key == Qt::Key\_S && (it->x == b&&it->y>=800&&it->y<=1100)) {

it = notes.erase(it); // 移除音符

score++;

} else {

if (key == Qt::Key\_K && (it->x == c&&it->y>=800&&it->y<=1100)) {

it = notes.erase(it); // 移除音符

score++;

}

else{

if (key == Qt::Key\_L &&(it->x == d&&it->y>=800&&it->y<=1100)) {

it = notes.erase(it); // 移除音符

score++;

}

else{

++it;

} } } }

update(); // 触发重绘 }

添加按钮，实现游戏的暂停与继续。

void gamewindow::**on\_pushButton\_clicked**()

{

if(ispaused){

musicplayer->play();

gameTimer->start();

ispaused=false;

}

else

{ musicplayer->pause();

gameTimer->stop();

ispaused=true;

} }//暂停

1. **单元测试与常见问题**
2. 每完成一个阶段的代码编写需进行一次单元测试；在制作谱面时，没完成一个乐句也应进行一次测试，以不断调整。
3. 在根据歌曲文件和乐谱的音符时值编写谱面时，经常出现下落时间与旋律不匹配的情况，时快时慢，此时需要根据测试情况调整音符时值。
4. 在编写谱面时要注意同一轨道不能连续出现多个音符，否则会影响游戏体验，同时在键盘交互时可能会错误地一次性移除多个音符。可以根据旋律的上行和下行决定对应轨道，同时注重游戏的体验感。
5. **收获**

通过本项目，学习了Qt的基本使用方法，包括信号槽、ui设计以及Qtimer、Qpainter等常用插件，同时对C++有了更深刻的了解，提升了发现问题和解决问题的能力。

1. 控件与信号槽

在创建控件时一般有两种方法，一是在ui界面找到所需控件拖动至页面中，右键直接连接信号槽；二是在代码文件中通过手动编写代码创建控件、修改参数和槽函数。前者较为简便，但能够实现的功能有限；后者需要的操作较复杂，但自由度更高。

1. 音乐播放

在添加音乐时，由于所使用的Qt版本较低，无法直接调用QMediaPlayer插件。在查询软件说明后了解到可以通过在主函数中import Qtmultimedia插入Qmusicplayer插件，从而实现音乐播放。

1. 计时器

计时是本项目核心逻辑代码中必不可少的环节，大部分函数都需根据时间进行更新或者重绘。在查阅资料后，知道了QTimer的声明与定义方法，从而解决了这一问题。

1. 初始化

在定义并初始化note和notedata结构体时，通过append关键词初始化列表，程序报错；研究后发现==的使用不规范，最后通过修改remove函数以及调整note的参数列表解决。