南開大學
Nankai University

高级语言程序设计

实验报告

南开大学 计算机科学与技术
姓名 范丁瑛
学号 2310887
班级 计算机卓越班
2024 年 5 月 14 日

目　录

高级语言程序设计大作业实验报告

一、作业题目

使用C++图形化完成音乐程序（射击类音乐游戏+电子钢琴）

二、开发软件

Qt6.7.0

三、 课题要求

（1）面向对象

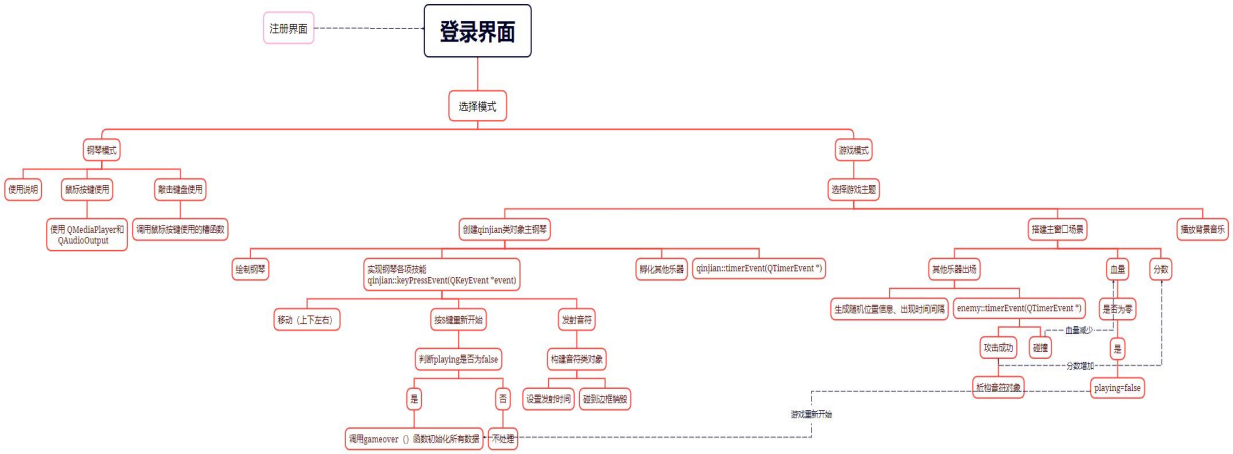
（2）单元测试

（3）模型部分

（4）验证

四、主要流程

（一）整体流程



（二）类视图

qinjian//游戏模式中的主钢琴类

bool playing=true;//判断游戏是否进行
QSoundEffect pianosound;//音效
"QGraphicsTextItem* messageItem=nullptr;//游戏结束时的提示行
WIDTH//主屏幕宽度
HEIGHT//主屏幕高度"
MusicPoint* musicpointplay//构建音符对象
qinjian(QGraphicsItem *parent=nullptr);//将钢琴在主屏中绘制出来
"virtual void keyPressEvent(QKeyEvent *event) override;//使用键盘进行操作（上下左右移动，空格键发射音符，S 键重新开始
void enemySpawn();//创造其他乐器孵化机

enemy//其他乐器类

enemy(QGraphicsItem *parent);
//绘制其他乐器
virtual void timerEvent(QTimerEvent*event) override;
//进行被攻击操作"

score//分数

int score=0;//设分数初始值
Score(QGraphicsItem *parent=nullptr);// 构造函数设置分数
void increase();//分数增加
void reset();//设置分数
static Score& getInstanance()//获取分数

MusicPoint//音符类

MusicPoint::MusicPoint(QGraphicsItem*parent):QGraphicsPixmapItem(parent)//创建音符
void MusicPoint::timerEvent(QTimerEvent *)//设置碰到边框销毁
setPos(x(),y()-change::musicpoinSpeed);// 出现速度位置

health 类//血量类

int health=change::HealthStart;
Health(QGraphicsItem *parent=nullptr);
//构造函数设置血量
int getHealth()//获取血量
void decrease();//血量减少函数
void reset();//设置血量函数
static Health& getInstanance(); //获取血量
int health=change::HealthStart;//设置初始血量

MainWindows 类//登录界面

MainWindow(QWidget *parent); //构造函数设置窗口信息
~MainWindow(); //析构函数
void on_btn_regis_clicked();//跳转到注册界面
void on_but_quit_clicked();//返回
void on_checkBox_clicked(bool checked);//显示密码
void on_btn_login_clicked();//登录，跳转到主界面
Ui::MainWindow *ui;//ui 界面

regis 类//注册界面

explicit regis(QWidget *parent = nullptr);//构造函数设置窗口信息
void on_btn_regis_clicked();//完成注册跳转到登录界面
void on_btn_quit_clicked();//返回
Ui::regis *ui;//ui 界面

platform 类//电子钢琴界面

使用鼠标实现各项功能：
void on_Button_C_clicked();
void on_Button_C_2_clicked();
void on_Button_D_clicked();
void on_Button_Eb_clicked();
void on_Button_E_clicked();
void on_Button_F_clicked();
void on_Button_F_2_clicked();
void on_Button_G_clicked();
void on_Button_G_2_clicked();
void on_Button_A_clicked();
void on_Button_Bb_clicked();
void on_Button_B_clicked();
void on_Button_C_3_clicked();
void on_Button_C_4_clicked();
void on_Button_D_2_clicked();
void on_Button_Eb_2_clicked();
void on_Button_E_2_clicked();
void keyPressEvent(QKeyEvent *event);//使用键盘使用
void keyReleaseEvent(QKeyEvent *event);

chooseplay 类//选择模式界面

void on_youxi_clicked();//游戏模式
void on_pushButton_2_clicked();//钢琴模式
void on_pushButton_clicked();//返回

background 类//选择游戏主题界面

void on_skyback_clicked();//跳转天空主题
void on_nightfallsky_clicked();//跳转黄昏主题
void on_riverback_clicked();//跳转河岸主题
void on_blueskyback_clicked();//跳转蓝天主题

introduction 类//电子琴使用说明界面

introduction(QWidget *parent = nullptr);
//构造函数显示使用说明

（三）核心代码及思路

1.登录界面绘制及部分功能

实现绘制、跳转、输入、显示（槽函数实现）、退出等功能

设置长度功能：setEchoMode(QLineEdit::Password)将输入的密码 1 变成密文，使其不可见，接着 setMaXLength 函数限定输入的密码不超过六位。

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("用户登录");
    ui->let_passwd->setEchoMode(QLineEdit::Password);//设置密码不可见
    ui->let_passwd->setMaxLength(6);//最大设置长度不超过6
}

//跳转到注册界面
void MainWindow::on_btn_regis_clicked()
{
    QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));
    regis *w=new regis();
    w->show();
    this->hide();
}

//退出
void MainWindow::on_but_quit_clicked()
{
    this->close();
}
```

显示功能实现：

通过槽函数，当显示按钮按下后，通过setEchoMode(QLineEdit::Normal);将密文变为可见

```
//显示
void MainWindow::on_checkBox_clicked(bool checked)
{
    if(checked)
    {
        //show
        ui->let_passwd->setEchoMode(QLineEdit::Normal);
    }
    else
    {
        ui->let_passwd->setEchoMode(QLineEdit::Password);//no show
    }
}
```

跳转主界面功能：新建主界面类指针类对象

```
//跳转主界面
void MainWindow::on_btn_login_clicked()
{
    QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));

    //创建场景
    chooseplay* v=new chooseplay;
    v->show();
    this->hide();
    this->close();
```

（注册界面、选择模式界面与登录界面大致相同，略）

2. 钢琴模式——电子钢琴功能实现

两种实现方式：鼠标点击琴键实现和直接敲击键盘实现。

（1）用鼠标点击琴键实现

调用槽函数并在槽函数中创建QMediaPlayer和QAudioOutput的指针对象

以中音区C（白键）和C#（黑键）为例

```
//中音
//C 523
void platform::on_Button_C_clicked()
{
    QMediaPlayer* bgm=new QMediaPlayer;
    QAudioOutput* audioOutput=new QAudioOutput();
    bgm->setAudioOutput(*&audioOutput);
    bgm->setSource(QUrl("qrc:/res/40-C  -s.wav"));
    bgm->play();
}

//C#    554
void platform::on_Button_C_2_clicked()
{
    QMediaPlayer* bgm=new QMediaPlayer;
    QAudioOutput* audioOutput=new QAudioOutput();
    bgm->setAudioOutput(*&audioOutput);
    bgm->setSource(QUrl("qrc:/res/p_41_cz4.mp3"));
    bgm->play();
}
```

（2）使用键盘，敲击键盘实现

通过直接调用上方使用鼠标点击键盘实现的槽函数，实现敲击键盘是电子钢琴发出声音。这里直接调用槽函数的方法可以大大减少我们的代码量，使代码更加简洁清晰。

以 W 和 S 键为例，分别是 C#和 D

```
case Qt::Key_W:emit ui->Button_C_2->click();break;
case Qt::Key_S:emit ui->Button_D->click();break;
```

3.游戏模式

在槽函数里首先需要首先先创建钢琴对象 qinjian* player=new qinjian;这里需要通过 setPixmap 把钢琴在屏幕中绘制出来。其次需要构建一个 QGraphicsView 指针对象 view 绘制游戏界面，使用一系列函数设置窗口属性，调用用 show()函数显示窗口。接着创建 QMediaPlayer 类和 QAudioOutput 类的指针播放背景音乐。最后调用 close()函数关闭此界面。

以“飞云之上”主题为例，即 skyback 按钮，另外三个按钮同理：

```
void background::on_skyback_clicked()
{
    qinjian* player=new qinjian;//创建钢琴对象
    //QResource::registerResource("./piano.rcc");

    //创建图标
    QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));

    //创建场景
    QGraphicsScene* scene=new QGraphicsScene();//gai
    //将琴键添加到场景中
    scene->addItem(player);
    scene->setSceneRect(0,0,690,388);//
    scene->setBackgroundBrush(QBrush(QImage(":/res/sky.jpg")));
    //创建分数文字item
    scene->addItem(&Score::getInstanance());
    scene->setStickyFocus(true);//不会在点击取消状态

    //创建分数文字item
    scene->addItem(&Health::getInstanance());
    //QGraphicsView view(scene);//gai
    QGraphicsView* view=new QGraphicsView(scene);//绘制
    view->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);//（发现图片大小不合适）去掉滚轮条
    view->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    view->setFixedSize(690,388);//大小
    view->setWindowTitle(TITLE);//名称
    view->show();
    QMediaPlayer* bgMusic=new QMediaPlayer();//播放背景音乐
    QAudioOutput* audioOutput=new  QAudioOutput();
    bgMusic->setAudioOutput(*&audioOutput);
    bgMusic->setSource(QUrl("qrc:/res/Pianoboy - The Truth That You Leave.mp3"));
    bgMusic->play();
    this->close();
}
```

其余三个按钮的功能及实现同上。

（1）选择主题界面

在槽函数里首先需要首先先创建钢琴对象 qinjian* player=new qinjian;这里需要通过 setPixmap 把钢琴在屏幕中绘制出来。其次需要构建一个 QGraphicsView 指针对象 view 绘制游戏界面，使用一系列函数设置窗口属性，调用用 show()函数显示窗口。接着创建 QMediaPlayer 类和 QAudioOutput 类的指针播放背景音乐。最后调用 close()函数关闭此界面。

以“飞云之上”主题为例，即 skyback 按钮，另外三个按钮同理：

```
void background::on_skyback_clicked()
{
    qinjian* player=new qinjian;//创建钢琴对象
    //QResource::registerResource("./piano.rcc");

    //创建图标
    QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));

    //创建场景
    QGraphicsScene* scene=new QGraphicsScene();//gai
    //将琴键添加到场景中
    scene->addItem(player);
    scene->setSceneRect(0,0,690,388);//
    scene->setBackgroundBrush(QBrush(QImage(":/res/sky.jpg")));
    //创建分数文字item
    scene->addItem(&Score::getInstanance());
    scene->setStickyFocus(true);//不会在点击取消状态

    //创建分数文字item
    scene->addItem(&Health::getInstanance());
    //QGraphicsView view(scene);//gai
    QGraphicsView* view=new QGraphicsView(scene);//绘制
    view->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);//（发现图片大小不合适）去掉滚轮条
    view->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    view->setFixedSize(690,388);//大小
    view->setWindowTitle(TITLE);//名称
    view->show();
    QMediaPlayer* bgMusic=new QMediaPlayer();//播放背景音乐
    QAudioOutput* audioOutput=new  QAudioOutput();
    bgMusic->setAudioOutput(*&audioOutput);
    bgMusic->setSource(QUrl("qrc:/res/Pianoboy - The Truth That You Leave.mp3"));
    bgMusic->play();
    this->close();
}
```

其余三个按钮的功能及实现同上。

（2）主钢琴上下左右移动功能实现

通过#include<QKeyEvent>头文件控制键盘，其中需要注意控制左右移动的范围及缩放比例，防止钢琴移动范围超过主屏幕。

```
switch(event->key())
{
case Qt::Key_Left:
    if(pos().x()>0)
    {
        setPos(x()-pianoMove,y());//钢琴左移速度
    }
    break;
case Qt::Key_Right:
    if(pos().x()<WIDTH-boundingRect().width()*0.2)//控制右边移动范围，注意缩放比例
    {
        setPos(x()+pianoMove,y());//钢琴右移速度
    }
    break;
case Qt::Key_Down:
    if(pos().y()<HEIGHT-boundingRect().width()*0.2)//控制下边移动范围，注意缩放比例
    {
        setPos(x(),y()+pianoMove);//钢琴下移速度
    }
    break;
case Qt::Key_Up://按上键
    if(pos().y()>0)
    {
        setPos(x(),y()-pianoMove);//钢琴上移速度
    }
    break;
```

（3）按空格键发射音符

首先需要创建音符对象的指针，同时我们需要一个 int 类型的变量记录音符释放的位置，通过 scene()->addItem(musicpointplay)将音符加入界面中。

```
case Qt::Key_Space:
    {
    pianosound.play();//播放音效
    MusicPoint* musicpointplay=new MusicPoint;//创建音符对象
    int temp=x()+boundingRect().width()*change::qinjianScale/2;//只考虑主钢琴宽度
    temp+=musicpointplay->boundingRect().width()*musicpointScale*1.5;//向右移动2个音符的宽度，将音符发射到中间
    musicpointplay->setPos(temp,y());
    scene()->addItem(musicpointplay);
    }
    break;
```

（4）按 S 键重新开始的实现

S 是按键重新开始，通过 playing 判断是否血量为零，保证结束游戏时才重新开始，将 playing 设置为 true，，血量分数恢复，使游戏继续进行。

```
case Qt::Key_S:
    if(playing)return;
    playing=true;
    Health::getInstanance().reset();
    Score::getInstanance().reset();
    messageItem->hide();
```

（5）攻击目标——其他乐器的创建

基本思路：在 qinjian 类中创建孵化函数，即 enemySpawn()函数将其他乐器构建出来，再在 enemy 类中通过随机函数生成随机位置的信息，使其随机出现在屏幕中的不同的位置。其中 int 型变量 max 控制随机数的最大范围，防止生成位置超出限度。运用列表 QList<QGraphicsItem *> itemList=collidingItems()将对象拿出来，判断音符与钢琴、其他乐器的相对位置，从而判断是否相撞或是否攻击成功，通过 if 语句实现两种不同情况的分支：如果攻击成功，调用分数增加函数 Score::getInstanance().increase()，使用 delete 关键字删除其他乐器和音符；如果相撞，调用血量减少的函数 Health::getInstanance().decrease()，同时删除音符。

在 qinjian 类中构建 enemySpwan（）函数孵化其他乐器

```
void qinjian::enemySpawn()
{
    if(playing)
    {
        enemy *enemyplay=new enemy;//创建小提琴攻击对象
        scene()->addItem(enemyplay);
    }
}
```

在 enemy 类中通过随机函数生成随机位置信息，同时通过 startTimer(change::enemyTimer)设置生成其他乐器的时间间隔

```
enemy::enemy(QGraphicsItem *parent):QGraphicsPixmapItem(parent)
{
    setPixmap(QPixmap(":/res/enemyviolin.png"));
    setScale(change::enemyScale);
    int max=WIDTH-boundingRect().width()*change::enemyScale;//其他乐器生成范围
    int randomNumber=QRandomGenerator::global()->bounded(1,max);//随机数生成
    setPos(randomNumber,0);//生成的位置信息
    startTimer(change::enemyTimer);
}
```

进行是否相撞是否攻击成功的判断

```
void enemy::timerEvent(QTimerEvent *)
{//运用列表将对象拿出来
    QList<QGraphicsItem *> itemList=collidingItems();
    for(auto item:itemList)
    {
        if(typeid(*item)==typeid(qinjian))
        {//撞到敌机，血量减少
            Health::getInstanance().decrease();//调用血量减少函数
            scene()->removeItem(this);
            delete this;
            return;
        }
        if(typeid(*item)==typeid(MusicPoint))
        {
            //需要加分
            Score::getInstanance().increase();
            scene()->removeItem(item);
            scene()->removeItem(this);
            delete item;
            delete this;//删掉其他乐器
            return;
        }//判断是否相撞
    }
    setPos(x(),y()+change::enemySpeed);
    if(y()>HEIGHT)
    {
        scene()->removeItem(this);
        delete this;
        return;
    }
}
```

（6）实现游戏状态判断

当 playing 是 true 时即为游戏可以进行，此时程序可以产生其他乐器，用户可以实现发射音符、移动钢琴等操作。

当 playing 被设置为 false 时即表示游戏结束，此时需要调用 gameOver()函数实现游戏结束的功能。

```
void qinjian::gameOver()
{
    playing=false;
    for(auto item:scene()->items())
    {
        if(typeid(*item)==typeid(enemy))
        {
            scene()->removeItem(item);
            delete item;//游戏结束删除所有其他乐器
        }
    }
    if(!messageItem)
    {
        messageItem=new QGraphicsTextItem;
        scene()->addItem(messageItem);
        QString message("Game Over!按s键重新开始! ");
        messageItem->setPlainText(message);
        messageItem->setDefaultTextColor(Qt::red);
        QFont font("Colorier New",change::FontSize*2,QFont::Bold);
        messageItem->setFont(font);
        QFontMetrics fm(font);
        int msgWidth=fm.horizontalAdvance(message);//水平方向跨度多少
        messageItem->setPos(WIDTH/2-msgWidth/2,HEIGHT/2);//居中
    }else
        messageItem->show();
}
```

通过 playing 及血量大小判断游戏是否结束

```
void qinjian::timerEvent(QTimerEvent *)
{
    if(playing)
    {
        enemySpawn();
    }
    if(Health::getInstanance().getHealth()<=0)
    {
        gameOver();//如果血量小于零，调用gameOver函数，结束游戏
    }
}
```

（7）实现血量的减少及分数的增加功能

1）血量减少功能

```
void Health::decrease()
{
    --health;
    setPlainText("血量"+QString::number(health));//更新文字显示
}

void Health::reset()
{
    health=change::HealthStart;
    setPlainText("血量"+QString::number(health));
    setDefaultTextColor(Qt::blue);
    setFont(QFont("Colorier New",change::FontSize,QFont::Bold));
    setPos(x(),change::FontSize*2);//健康值的显示在往下两个字符大小处
}
```

2）分数增加功能

```
void Score::reset()
{
    score=0;

    setPlainText("分 数: "+QString::number(score));
    setDefaultTextColor(Qt::green);
    setFont(QFont("Courier New",change::FontSize,QFont::Bold));//字体加粗
}
void Score::increase()
{
    ++score;
    setPlainText("分 数"+QString::number(score));
}
```

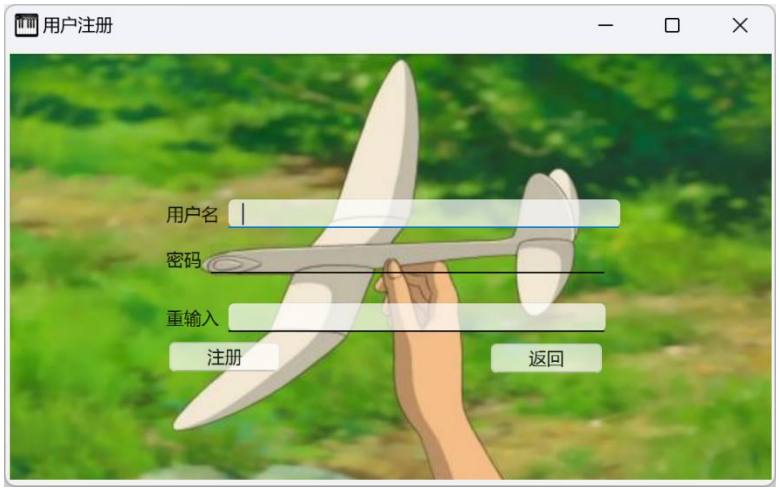
五、代码测试

1.登录界面

3.登陆界面的显示/不显示密码

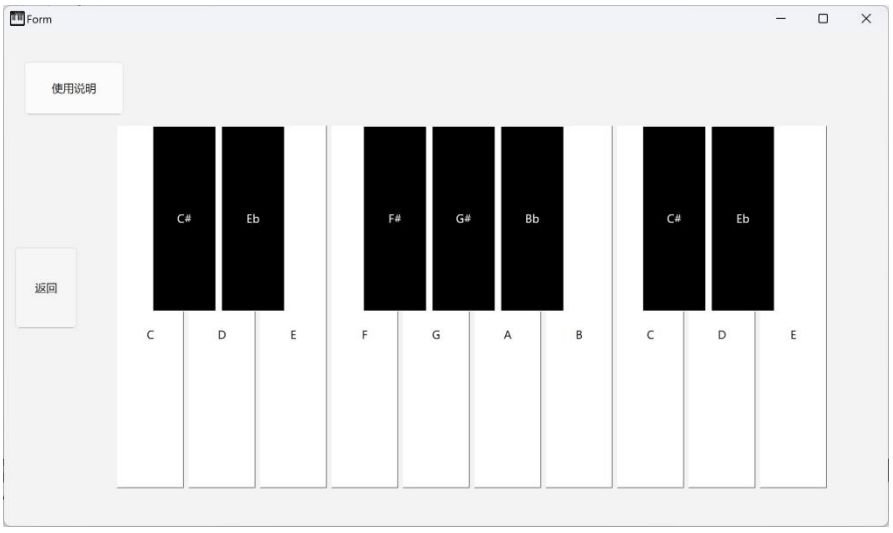
不显示：　　显示：

3.注册界面

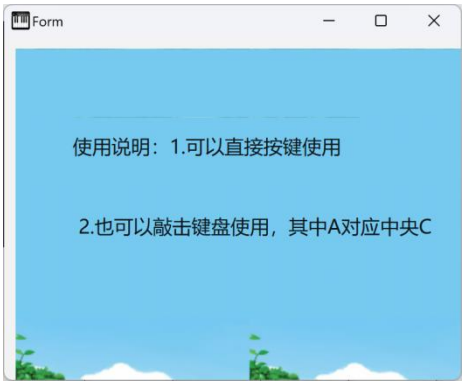


4.选择模式界面

5.钢琴模式界面

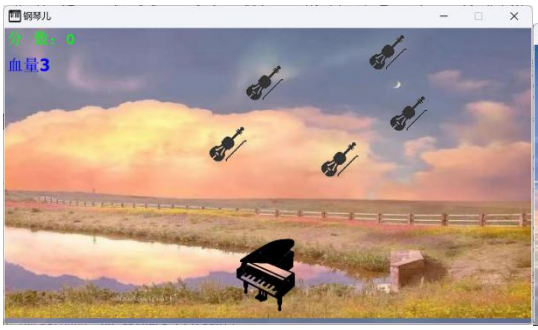


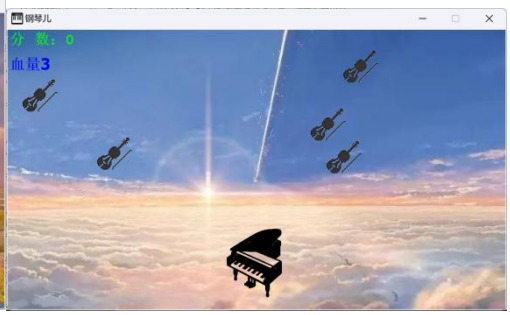
6. 钢琴模式下使用说明界面

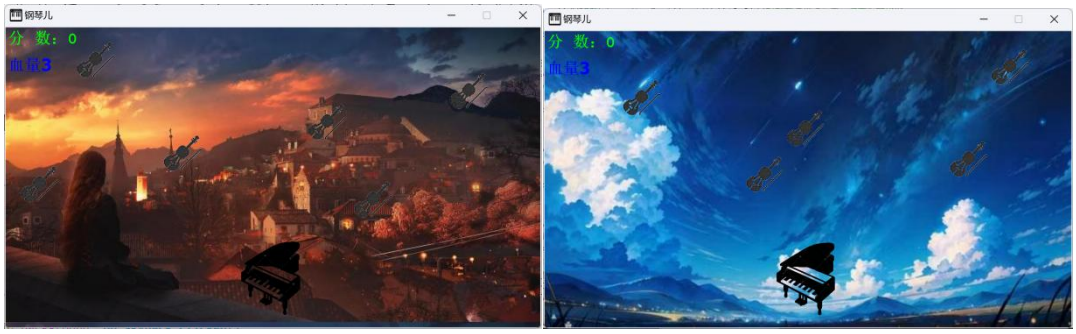


7. 选择游戏主题界面

8. 四种不同主题界面展示







9. 攻击成功，分数增加

10. 撞击其他乐器，血量减少

11. 游戏结束重新开始

六、遇到的困难及解决办法

1.Qt 安装时的第一道关卡。

自己因为心急误下载了商业版，七七八八填了很多信息之后还不能用，折腾了很久才安装成功。安装完成后发现自己下载错了，试用期十天之后就无法使用，在网上下载安装包却缺少套件，最后只能连夜给 qt 官方写邮件。Qt 官方回复很及时，但是停止了我的商业许可之后仍旧无法正常安装开源版 qt（期间我也在 csdn 以及 bilibili 等网站上查询可能的问题或解决办法，也尝试了各种镜像安装，最后终于在折腾了一周之后找到了一个合适的安装包(期间我应该大大小小安装了十几二十次 qt 的安装包）。

2.图片无法加载（版本不同问题）

自己曾经查资料、看网上教程找插图片办法，可是将所有的方法（ui、Qgraphics 等）尝试了很多遍仍无法解决，期间我曾想是否是我的路径不对？图片格式不对？二进制文件没转成功……将自己的鼠标置于代码上可以清晰看到图片的加载，说明并非上述与图片有关问题，而是代码出错（首先显示之前的方法版本不兼容，最后多次尝试后发现自己因尝试过多方法代码删除不完全导致无法实现）。

3.音频无法播放出来

自己在网上查阅多种资料，尝试 QMediaPlay、QSound、QSoundEffect、QMediaPlayer 等多种头文件仍无法播放，最后发现是路径复制出来

问题，绝对路径 vs 相对路径，应选择 Copy URL 而非 Path 路径。

4.制作页面跳转功能时忘记将变量开辟在堆区

之前已在主函数里运用绘图等各种工具了，自己想出利用全局变量+判断，将实现 ui 与主函数间页面跳转，但在实现中出现了错误使用变量 a 的问题（变量 a 在主函数一开始已定义）报了 main.cpp:42:8: Value of type 'QApplication' is not contextually convertible to 'bool'的错，且 qt 貌似不支持 bool 与 int 的隐式转化

最后将原来写在主函数中的与界面和背景音乐有关的全部写进槽函数里，一开始界面总是一闪而过，最后发现应该是需要将所有的内容放进堆区里，否则类执行结束对象就会被析构。

七、收获

（一）加深了对 C++面向对象编程的理解。

（二）养成了写代码量较大的项目时随手写注释的习惯。

（三）培养了自己遇到相关问题动手去查询的习惯。

（四）学会了 C++图形化编程中图形绘制函数的使用。

要加入头文件#include<QGraphicsScene>

1.生成图形窗口

```
QGraphicsView* view=new QGraphicsView(scene);//绘制
view->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);//（发现图片大小不合适）去掉滚轮条
view->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
view->setFixedSize(690,388);//大小
view->setWindowTitle(TITLE);//名称
view->show();
```

创建图形窗口QGraphicsView指针

设置长度宽度等数据

2.创建图标

```
QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));
```

3.调整大小及设置名称

```
view->setFixedSize(690,388);//大小
view->setWindowTitle(TITLE);//名称
```

4.文字输出

```
messageItem=new QGraphicsTextItem;
scene()->addItem(messageItem);
QString message("Game Over!按s键重新开始! ");
messageItem->setPlainText(message);
messageItem->setDefaultTextColor(Qt::red);
QFont font("Colorier New",change::FontSize*2,QFont::Bold);
messageItem->setFont(font);
QFontMetrics fm(font);
int msgWidth=fm.horizontalAdvance(message);//水平方向跨度多少
messageItem->setPos(WIDTH/2-msgWidth/2,HEIGHT/2);//居中
```

WIDTH：宽度

HEIGHT：高度

QString message设置输出的文字

messageItem->setDefaultTextColor(Qt::red)设置字体颜色

QFont font("Colorier New",change::FontSize*2,QFont::Bold)；调整为加粗

messageItem->setPos(WIDTH/2-msgWidth/2,HEIGHT/2);居中

5.贴图操作

（1）包含头文件 QGraphicsScene

#include<QGraphicsScene>

（2）创建资源文件

将所需的图片资源放到一个文件夹中，创建资源文件 qrc 并更改名称。创建好资源文件后右键 qrc 文件选择“open in editor”添加前缀并添加文件

（3）载入图片

```
setPixmap(QPixmap(":/res/pianoplay.png"));
```

":/res/pianoplay.png"为图片的路径

（4）输出图片并调整图片

```
setPixmap(QPixmap(":/res/pianoplay.png"));
setScale(0.2);//设置图标倍数
setPos(WIDTH/2-boundingRect().width()*0.2/2,HEIGHT-boundingRect().height()*0.2);//设置在底部正中间
setFlag(QGraphicsItem::ItemIsFocusable);
setFocus();
```

WIDTH：图片的宽度

HEIGHT：图片的高度

setFocus()：将图片放置在中央

（五）了解了 Qt 中读取键鼠信息的代码实现方式。

1. 键盘

通过#include<QKeyEvent>头文件，switch 语句实现不同键盘的使用，如：

```
case Qt::Key_Space:
    {
    pianosound.play();//播放音效
    MusicPoint* musicpointplay=new MusicPoint;//创建音符对象
    int temp=x()+boundingRect().width()*change::qinjianScale/2;//只考虑主钢琴宽度
    temp+=musicpointplay->boundingRect().width()*musicpointScale*1.5;//向右移动2个音符的宽度，将音符发射到中间
    musicpointplay->setPos(temp,y());
    scene()->addItem(musicpointplay);
    }
    break;
```

```
case Qt::Key_W:emit ui->Button_C_2->click();break;
case Qt::Key_S:emit ui->Button_D->click();break;
```

2. 鼠标

通过槽函数进行操作

```
void platform::on_Button_C_clicked()
{
    QMediaPlayer* bgm=new QMediaPlayer;
    QAudioOutput* audioOutput=new QAudioOutput();
    bgm->setAudioOutput(*&audioOutput);
    bgm->setSource(QUrl("qrc:/res/40-C  -s.wav"));
    bgm->play();
}
```

（六）了解了 Qt 编程中 BGM 的插入与播放。

如背景音乐的播放：

```
    QMediaPlayer* bgMusic=new QMediaPlayer();//播放背景音乐
    QAudioOutput* audioOutput=new  QAudioOutput();
    bgMusic->setAudioOutput(*&audioOutput);
    bgMusic->setSource(QUrl("qrc:/res/Pianoboy - The Truth That You Leave.mp3"));
    bgMusic->play();
```

（七）了解 ui 界面各种功能及槽函数使用

了解了 ui 界面中插入标签、按钮、输入的文本框的各项美化界面。按钮跳转功能的实现需要将按钮转到槽，在槽函数中新建指向下一窗口的指针。

```
//跳转到注册界面
void MainWindow::on_btn_regis_clicked()
{
    QApplication::setWindowIcon(QIcon(":/res/tubiao.ico"));
    regis *w=new regis();
    w->show();
    this->hide();
}

//退出
void MainWindow::on_but_quit_clicked()
{
    this->close();
}
```