南开大学 计算机大类

姓名 沈超

学号 2213404

班级 工科试验班模拟一班

2023年5月14日

高级语言程序设计

实验报告

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． 整体流程](#_Toc20452)

[（1） 建立项目](#_Toc22508)

[（2） 搭建主场景](#_Toc28665)

[（3） 创建config.h头文件保存配置数据，便于后期维护与修改](#_Toc32428)

[（4） 在MainWindow中写出游戏运行中所有事件的成员函数](#_Toc23860)

[（5） 在MainWindow中写出游戏运行中所有事件的成员函数](#_Toc13349)

（6） 在main函数中注册二进制rcc文件

（7） 创建地图文件和类实现地图滚动

（8）创建提琴文件和类

（9）制作提琴发射出的音符文件和类

（10）创建休止符文件和类

（11）在MainWindow。Cpp中实现上述所有功能并创建检测碰撞函数

（12）创建休止符逐渐消失的文件和类

（13）在MainWindow.h中加入音符消失数组并设置音符消失对象信息

[五. 收获](#_Toc23116)

高级语言程序设计大作业实验报告

1. **作业题目**

碰撞的音符小游戏

1. **开发软件**

QT6 6.2.4

1. **课题要求**
2. 实现循环滚动的地图背景
3. 实现提琴随鼠标移动
4. 制作发射的音符及实现它们的移动
5. 制作射击目标的休止符
6. 进行音符与休止符发生碰撞的检测
7. 实现休止符碰到音符后逐渐变小消失的效果
8. **主要流程**
   1. **整体流程**

实现思路：

1）建立项目

2）搭建主场景

初始化窗口尺寸和标题

1. 创建config.h头文件保存配置数据，便于后期维护与修改

#ifndef CONFIG\_H

#define CONFIG\_H

/\*\*\*\*\*\*\*\*\*\* 游戏配置数据 \*\*\*\*\*\*\*\*\*\*/

#define GAME\_WIDTH 631 //宽度

#define GAME\_HEIGHT 760//高度

#define GAME\_TITLE "碰撞的音符" //标题

#define GAME\_RES\_PATH "./plane.rcc" //rcc文件路径

#define GAME\_ICON ":/res/violin.jpg"

/\*\*\*\*\*\*\*\*\*\* 地图配置数据 \*\*\*\*\*\*\*\*\*\*/

#define MAP\_PATH ":/res/piano2.png" //地图图片路径

#define MAP\_SCROLL\_SPEED 2 //地图滚动速度

#define GAME\_RATE 10 //刷新间隔，帧率 单位毫秒

#define HERO\_PATH ":/res/violinn.png"

/\*\*\*\*\*\*\*\*\*\* 音符配置数据 \*\*\*\*\*\*\*\*\*\*/

#define BULLET\_PATH ":/res/321.png" //音符图片路径

#define BULLET\_SPEED 5 //音符移动速度

#define BULLET\_NUM 30 //弹匣中音符总数

#define BULLET\_INTERVAL 20 //发射音符时间间隔

/\*\*\*\*\*\*\*\*\*\* 休止符配置数据 \*\*\*\*\*\*\*\*\*\*/

#define ENEMY\_PATH ":/res/01.png" //休止符资源图片

#define ENEMY\_SPEED 5 //休止符移动速度

#define ENEMY\_NUM 20 //休止符总数量

#define ENEMY\_INTERVAL 30 //休止符出场时间间隔

/\*\*\*\*\*\*\*\*\*\* 音符消失配置数据 \*\*\*\*\*\*\*\*\*\*/

#define BOMB\_PATH ":/res/0%1.png" //音符消失资源图片

#define BOMB\_NUM 20 //音符消失数量

#define BOMB\_MAX 7 //音符消失图片最大索引

#define BOMB\_INTERVAL 20 //音符消失切图时间间隔

#endif // CONFIG\_H

1. 在MainWindow中写出游戏运行中所有事件的成员函数

包括：初始化游戏场景启动游戏的initScene、实现各元素坐标更新的updatePosition、绘制提琴 音符 休止符 及休止符消失效果的paintEvent、实现鼠标移动完成效果的mouseMoveEvent、实现休止符出场的enemyToScene、检测音符与休止符是否发生碰撞的collisionDetection、以及对、游戏内各元素的初始化。

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<QTimer>

#include "map.h"

#include "heroplane.h"

#include "buller.h"

#include "enemyplane.h"

#include "bomb.h"

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void initScene();

//启动游戏 用于启动定时器对象

void playGame();

//更新坐标

void updatePosition();

//绘图事件

void paintEvent(QPaintEvent \*event);

//鼠标移动事件

void mouseMoveEvent(QMouseEvent \*);

//休止符出场

void enemyToScene();

//碰撞检测

void collisionDetection();

//休止符数组

EnemyPlane m\_enemys[ENEMY\_NUM];

//休止符出场间隔记录

int m\_recorder;

//地图对象

Map m\_map;

//提琴对象

HeroPlane m\_hero;

//定时器

QTimer m\_Timer;

//音符消失数组

Bomb m\_bombs[BOMB\_NUM];

//测试音符代码

Buller temp\_bullet;

};

#endif // MAINWINDOW\_H

1. 导入游戏内的图片资源并创建rcc文件、添加Qt资源文件
2. 在main函数中注册二进制rcc文件

#define GAME\_RES\_PATH "./music.rcc"

#include "mainwindow.h"

#include <QApplication>

#include <QResource>

#include "config.h"

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

{

QApplication a(argc, argv);

//注册外部的二进制资源文件

QResource::registerResource(GAME\_RES\_PATH);

MainWindow w;

w.show();

return a.exec();

}

1. 创建地图文件和类实现地图滚动

#ifndef MAP\_H

#define MAP\_H

#include <QPixmap>

class Map

{

public:

//构造函数

Map();

//地图滚动坐标计算

void mapPosition();

public:

//地图图片对象

QPixmap m\_map1;

QPixmap m\_map2;

//地图Y轴坐标

int m\_map1\_posY;

int m\_map2\_posY;

//地图滚动幅度

int m\_scroll\_speed;

};

#endif // MAP\_H

//cpp

#include "map.h"

#include "config.h"

Map::Map()

{

//初始化加载地图对象

m\_map1.load(MAP\_PATH);

m\_map2.load(MAP\_PATH);

//设置地图其实y轴坐标

m\_map1\_posY = -GAME\_HEIGHT;

m\_map2\_posY = 0;

//设置地图滚动速度

m\_scroll\_speed = MAP\_SCROLL\_SPEED;

}

void Map::mapPosition()

{

//处理第一张图片滚动

m\_map1\_posY += MAP\_SCROLL\_SPEED;

if(m\_map1\_posY >= 0)

{

m\_map1\_posY =-GAME\_HEIGHT;

}

//处理第二张图片滚动

m\_map2\_posY += MAP\_SCROLL\_SPEED;

if(m\_map2\_posY >= GAME\_HEIGHT )

{

m\_map2\_posY =0;

}

}

1. 创建提琴文件和类

#ifndef HEROPLANE\_H

#define HEROPLANE\_H

#include <QPixmap>

#include <QRect>

#include "buller.h"

#include <ctime>

class HeroPlane

{

public:

HeroPlane();

//发射音符

void shoot();

//设置提琴位置

void setPosition(int x, int y);

public:

//提琴资源 对象

QPixmap m\_Plane;

//提琴坐标

int m\_X;

int m\_Y;

//提琴的矩形边框

QRect m\_Rect;

//弹匣

Buller m\_bullets[BULLET\_NUM];

//发射间隔记录

int m\_recorder;

};

#endif // HEROPLANE\_H

//cpp

#include "heroplane.h"

#include "config.h"

HeroPlane::HeroPlane()

{

//初始化加载提琴图片资源

m\_Plane.load(HERO\_PATH);

//初始化坐标

m\_X = GAME\_WIDTH \* 0.5 - m\_Plane.width()\*0.5;

m\_Y = GAME\_HEIGHT- m\_Plane.height();

//初始化矩形框

m\_Rect.setWidth(m\_Plane.width());

m\_Rect.setHeight(m\_Plane.height());

m\_Rect.moveTo(m\_X,m\_Y);

//初始化发射间隔记录

m\_recorder = 0;

}

void HeroPlane::setPosition(int x, int y)

{

m\_X = x;

m\_Y = y;

m\_Rect.moveTo(m\_X,m\_Y);

}

void HeroPlane::shoot()

{

//累加时间间隔记录变量

m\_recorder++;

//判断如果记录数字 未达到发射间隔，直接return

if(m\_recorder < BULLET\_INTERVAL)

{

return;

}

//到达发射时间处理

//重置发射时间间隔记录

m\_recorder = 0;

//发射音符

for(int i = 0 ; i < BULLET\_NUM;i++)

{

//如果是空闲的音符进行发射

if(m\_bullets[i].m\_Free)

{

//将改音符空闲状态改为假

m\_bullets[i].m\_Free = false;

//设置发射的音符坐标

m\_bullets[i].m\_X = m\_X + m\_Rect.width()\*0.5 - 10;

m\_bullets[i].m\_Y = m\_Y - 25 ;

break;

}

}

}

1. 制作提琴发射出的音符文件和类

#ifndef BULLET\_H

#define BULLET\_H

#include "config.h"

#include <QPixmap>

class Buller

{

public:

Buller();

//更新音符坐标

void updatePosition();

public:

//音符资源对象

QPixmap m\_Buller;

//音符坐标

int m\_X;

int m\_Y;

//音符移动速度

int m\_Speed;

//音符是否闲置

bool m\_Free;

//音符的矩形边框（用于碰撞检测）

QRect m\_Rect;

};

#endif // BULLET\_H

//cpp

#include "buller.h"

Buller::Buller()

{

//加载音符资源

m\_Buller.load(BULLET\_PATH);

//音符坐标 初始坐标可随意设置，后期会重置

m\_X = GAME\_WIDTH\*0.5 - m\_Buller.width()\*0.5;

m\_Y = GAME\_HEIGHT;

//音符状态

m\_Free = true;

//音符速度

m\_Speed = BULLET\_SPEED;

//音符矩形框

m\_Rect.setWidth(m\_Buller.width());

m\_Rect.setHeight(m\_Buller.height());

m\_Rect.moveTo(m\_X,m\_Y);

}

void Buller::updatePosition()

{

//如果音符是空闲状态，不需要坐标计算

//玩家提琴可以控制音符的空闲状态为false

if(m\_Free)

{

return;

}

//音符向上移动

m\_Y -= m\_Speed;

m\_Rect.moveTo(m\_X,m\_Y);

if(m\_Y <= -m\_Rect.height())

{

m\_Free = true;

}

}

1. 创建休止符文件和类

#ifndef ENEMYPLANE\_H

#define ENEMYPLANE\_H

#include <QPixmap>

class EnemyPlane

{

public:

EnemyPlane();

//更新坐标

void updatePosition();

public:

//休止符资源对象

QPixmap m\_enemy;

//位置

int m\_X;

int m\_Y;

//休止符的矩形边框（碰撞检测）

QRect m\_Rect;

//状态

bool m\_Free;

//速度

int m\_Speed;

};

#endif // ENEMYPLANE\_H

//cpp

#include "enemyplane.h"

#include "config.h"

EnemyPlane::EnemyPlane()

{

//休止符资源加载

m\_enemy.load(ENEMY\_PATH);

//休止符位置

m\_X = 0;

m\_Y = 0;

//休止符状态

m\_Free = true;

//休止符速度

m\_Speed = ENEMY\_SPEED;

//休止符矩形

m\_Rect.setWidth(m\_enemy.width());

m\_Rect.setHeight(m\_enemy.height());

m\_Rect.moveTo(m\_X,m\_Y);

}

void EnemyPlane::updatePosition()

{

//空闲状态，不计算坐标

if(m\_Free)

{

return;

}

m\_Y += m\_Speed;

m\_Rect.moveTo(m\_X,m\_Y);

if(m\_Y >= GAME\_HEIGHT + m\_Rect.height())

{

m\_Free = true;

}

}

1. 在MainWindow。Cpp中实现上述所有功能并创建检测碰撞函数

#include "mainwindow.h"

#include "config.h"

#include <QIcon>

#include <QPainter>

#include <QMouseEvent>

#include "buller.h"

#include <ctime>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

{

//初始化场景

initScene();

//启动游戏

playGame();

}

MainWindow::~MainWindow()

{

}

void MainWindow::initScene()

{

//初始化窗口大小

setFixedSize(GAME\_WIDTH,GAME\_HEIGHT);

//设置窗口标题

setWindowTitle(GAME\_TITLE);

//设置图标资源

setWindowIcon(QIcon( GAME\_ICON)); //加头文件 #include <QIcon>

//定时器设置

m\_Timer.setInterval(GAME\_RATE);

m\_recorder = 0;

//随机数种子

srand((unsigned int)time(NULL));

}

void MainWindow::playGame()

{

//启动定时器

m\_Timer.start();

//监听定时器

connect(&m\_Timer,&QTimer::timeout,[=](){

//休止符出场

enemyToScene();

//更新游戏中元素的坐标

updatePosition();

//重新绘制图片

update();

//碰撞检测

collisionDetection();

});

}

void MainWindow::updatePosition()

{

//更新地图坐标

m\_map.mapPosition();

//发射音符

m\_hero.shoot();

//计算音符坐标

for(int i = 0 ;i < BULLET\_NUM;i++)

{

//如果音符状态为非空闲，计算发射位置

if(!m\_hero.m\_bullets[i].m\_Free)

{

m\_hero.m\_bullets[i].updatePosition();

}

}

//休止符坐标计算

for(int i = 0 ; i< ENEMY\_NUM;i++)

{

//非空闲休止符 更新坐标

if(m\_enemys[i].m\_Free == false)

{

m\_enemys[i].updatePosition();

}

}

//计算音符消失播放的图片

for(int i = 0 ; i < BOMB\_NUM;i++)

{

if(m\_bombs[i].m\_Free == false)

{

m\_bombs[i].updateInfo();

}

}

//测试音符代码

temp\_bullet.m\_Free = false;

temp\_bullet.updatePosition();

}

void MainWindow::paintEvent(QPaintEvent \*)

{

QPainter painter(this);

//绘制地图

painter.drawPixmap(0,m\_map.m\_map1\_posY , m\_map.m\_map1);

painter.drawPixmap(0,m\_map.m\_map2\_posY , m\_map.m\_map2);

//绘制提琴

painter.drawPixmap(m\_hero.m\_X,m\_hero.m\_Y,m\_hero.m\_Plane);

//绘制音符

for(int i = 0 ;i < BULLET\_NUM;i++)

{

//如果音符状态为非空闲，计算发射位置

if(!m\_hero.m\_bullets[i].m\_Free)

{

painter.drawPixmap(m\_hero.m\_bullets[i].m\_X,m\_hero.m\_bullets[i].m\_Y,m\_hero.m\_bullets[i].m\_Buller);

}

}

//绘制休止符

for(int i = 0 ; i< ENEMY\_NUM;i++)

{

if(m\_enemys[i].m\_Free == false)

{

painter.drawPixmap(m\_enemys[i].m\_X,m\_enemys[i].m\_Y,m\_enemys[i].m\_enemy);

}

}

//绘制音符消失图片

for(int i = 0 ; i < BOMB\_NUM;i++)

{

if(m\_bombs[i].m\_Free == false)

{

painter.drawPixmap(m\_bombs[i].m\_X,m\_bombs[i].m\_Y,m\_bombs[i].m\_pixArr[m\_bombs[i].m\_index]);

}

}

//测试音符代码

painter.drawPixmap(temp\_bullet.m\_X,temp\_bullet.m\_Y,temp\_bullet.m\_Buller);

}

void MainWindow::mouseMoveEvent(QMouseEvent \*event)

{

int x = event->x() - m\_hero.m\_Rect.width()\*0.5; //鼠标位置 - 飞机矩形的一半

int y = event->y() - m\_hero.m\_Rect.height()\*0.5;

//边界检测

if(x <= 0 )

{

x = 0;

}

if(x >= GAME\_WIDTH - m\_hero.m\_Rect.width())

{

x = GAME\_WIDTH - m\_hero.m\_Rect.width();

}

if(y <= 0)

{

y = 0;

}

if(y >= GAME\_HEIGHT - m\_hero.m\_Rect.height())

{

y = GAME\_HEIGHT - m\_hero.m\_Rect.height();

}

m\_hero.setPosition(x,y);

}

void MainWindow::enemyToScene()

{

m\_recorder++;

if(m\_recorder < ENEMY\_INTERVAL)

{

return;

}

m\_recorder = 0;

for(int i = 0 ; i< ENEMY\_NUM;i++)

{

if(m\_enemys[i].m\_Free)

{

//休止符空闲状态改为false

m\_enemys[i].m\_Free = false;

//设置坐标

m\_enemys[i].m\_X = rand() % (GAME\_WIDTH - m\_enemys[i].m\_Rect.width());

m\_enemys[i].m\_Y = -m\_enemys[i].m\_Rect.height();

break;

}

}

}

void MainWindow::collisionDetection()

{

//遍历所有非空闲的休止符

for(int i = 0 ;i < ENEMY\_NUM;i++)

{

if(m\_enemys[i].m\_Free)

{

//空闲休止符 跳转下一次循环

continue;

}

//遍历所有 非空闲的音符

for(int j = 0 ; j < BULLET\_NUM;j++)

{

if(m\_hero.m\_bullets[j].m\_Free)

{

//空闲音符 跳转下一次循环

continue;

}

//如果音符矩形框和休止符矩形框相交，发生碰撞，同时变为空闲状态即可

if(m\_enemys[i].m\_Rect.intersects(m\_hero.m\_bullets[j].m\_Rect))

{

m\_enemys[i].m\_Free = true;

m\_hero.m\_bullets[j].m\_Free = true;

//播放音符消失效果

for(int k = 0 ; k < BOMB\_NUM;k++)

{

if(m\_bombs[k].m\_Free)

{

//音符消失状态设置为非空闲

m\_bombs[k].m\_Free = false;

//更新坐标

m\_bombs[k].m\_X = m\_enemys[i].m\_X;

m\_bombs[k].m\_Y = m\_enemys[i].m\_Y;

break;

}

}

}

}

}

}

1. 创建休止符逐渐消失的文件和类

#ifndef BOMB\_H

#define BOMB\_H

#include "config.h"

#include <QPixmap>

#include <QVector>

class Bomb

{

public:

Bomb();

//更新信息（播放图片下标、播放间隔）

void updateInfo();

public:

//放休止符消失资源数组

QVector<QPixmap> m\_pixArr;

//休止符消失位置

int m\_X;

int m\_Y;

//休止符消失状态

bool m\_Free;

//休止符消失切图的时间间隔

int m\_Recoder;

//休止符消失时加载的图片下标

int m\_index;

};

#endif // BOMB\_H

//cpp

#include "bomb.h"

Bomb::Bomb()

{

//初始化音符消失图片数组

for(int i = 1 ;i <= BOMB\_MAX ;i++)

{

QString str = QString(BOMB\_PATH).arg(i);

m\_pixArr.push\_back(QPixmap(str));

}

//初始化坐标

m\_X = 0;

m\_Y = 0;

//初始化空闲状态

m\_Free = true;

//当前播放图片下标

m\_index = 0;

//音符消失间隔记录

m\_Recoder = 0;

}

void Bomb::updateInfo()

{

//空闲状态

if(m\_Free)

{

return;

}

m\_Recoder++;

if(m\_Recoder < BOMB\_INTERVAL)

{

//记录音符消失间隔未到，直接return，不需要切图

return;

}

//重置记录

m\_Recoder = 0;

//切换音符消失播放图片

m\_index++;

//注：数组中的下标从0开始，最大是6

//如果计算的下标大于6，重置为0

if(m\_index > BOMB\_MAX-1)

{

m\_index = 0;

m\_Free = true;

}

}

1. 在MainWindow.h中加入音符消失数组并设置音符消失对象信息

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<QTimer>

#include "map.h"

#include "heroplane.h"

#include "buller.h"

#include "enemyplane.h"

#include "bomb.h"

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void initScene();

//启动游戏 用于启动定时器对象

void playGame();

//更新坐标

void updatePosition();

//绘图事件

void paintEvent(QPaintEvent \*event);

//鼠标移动事件

void mouseMoveEvent(QMouseEvent \*);

//休止符出场

void enemyToScene();

//碰撞检测

void collisionDetection();

//休止符数组

EnemyPlane m\_enemys[ENEMY\_NUM];

//休止符出场间隔记录

int m\_recorder;

//地图对象

Map m\_map;

//提琴对象

HeroPlane m\_hero;

//定时器

QTimer m\_Timer;

//音符消失数组

Bomb m\_bombs[BOMB\_NUM];

//测试音符代码

Buller temp\_bullet;

};

#endif // MAINWINDOW\_H

**测试一切正常**

1. **收获**

掌握了分解大内容，逐个解决各对象问题，并对各对象运行进行协调

掌握了无限滚动地图的实现方法、目标逐渐消失的实现方法