南開大學
Nankai University

计算机学院

高级语言程序设计实验报告

小游戏：进击的小鸟

姓名：聂嘉欣

学号：2213254

专业：计算机科学与技术

2024 年 5 月 14 日

目录

1 作业题目

小游戏：《进击的小鸟》

2 开发软件

QT 5.9.9

3 课题要求

1）面向对象
2）单元测试
3）模型部分
4）验证

4 主要流程

本游戏实现思路如下：

4.1 主要游戏界面实现

主要游戏界面实现思路为：使用 QT 的窗口设计功能，定义了一个窗体类 “Bird”。
Bird 类中主要的函数如下：

```
3   Bird::Bird(QWidget *parent)
4       : QMainWindow(parent)
5       , ui(new Ui::Bird)
6   {
7       ui->setupUi(this);
8       this->setGeometry(QRect(600,300,480,480));
9       this->setWindowTitle("进击的小鸟！");
10      this->grabKeyboard();
11  }
```

图 4.1: Bird 的构造函数

图 4.1 构造函数中实现了构造窗体大小、窗体名称和捕获键盘事件。

```cpp
void Bird::paintEvent(QPaintEvent *event){
    QPainter painter(this);
    if(!blsRun){
        InitBird();
    }

    //画游戏背景
    QRect background(0,0,480,400);
    painter.drawPixmap(background,QPixmap(":/myImage/images/sky.png"));
    painter.setPen(Qt::white);
    painter.setBrush(Qt::white);
    QRect bottom(0,400,480,100);
    painter.drawRect(bottom);

    //画鸟
    painter.drawPixmap(vBird,QPixmap(":/myImage/images/bird.png"));//插入图片

    //画障碍
    for(int i=0;i<4;i++){
        painter.drawPixmap(barrier[i].down,QPixmap(":/myImage/images/barrier.png"));
        painter.drawPixmap(barrier[i].up,QPixmap(":/myImage/images/barrier.png"));
    }

    if(blsOver){
        timer1->stop();
        this->close();
        e=new End;
        e->manage(total_jump,total_jump);
        e->show();
    }
}
```

图 4.2: 重写 Bird 的 paintEvent 函数

在图 4.2 函数中，实现了对鸟的初始化、游戏背景的绘制、鸟的绘制、障碍的绘制以及游戏结束后的界面调用。

```cpp
void Bird::InitBird(){
    blsRun=true;
    QRect rect(80,160,40,40);
    vBird=rect;

    for(int i=0;i<4;i++){
        barrier[i].up.setBottom(0);
        barrier[i].down.setTop(480);
    }
    CreateBarrier();

    timer1=new QTimer(this);
    timer1->start(100);
    connect(timer1,SIGNAL(timeout()),SLOT(BirdJudge()));
    connect(timer1,SIGNAL(timeout()),SLOT(Bird_update()));

}
```

图 4.3: 初始化鸟的函数

在图 4.3 函数中，实现了对鸟的位置以及大小的定义、障碍的初始化（防止出现判断错误）以及

计时器的设置。

```
68  void Bird::keyPressEvent(QKeyEvent *event){
69      QKeyEvent *key=(QKeyEvent*)event;
70      switch(key->key()){
71      case Qt::Key_Up:nDirection=1;
72          break;
73      case Qt::Key_Down:nDirection=2;
74          break;
75      default:;
76      }
77  }
78
79  void Bird::keyReleaseEvent(QKeyEvent *event){
80      QKeyEvent *key=(QKeyEvent*)event;
81      switch(key->key()){
82      case Qt::Key_Up:nDirection=0;
83          break;
84      case Qt::Key_Down:nDirection=0;
85          break;
86      default:;
87      }
88  }
```

图 4.4: 键盘事件函数

图 4.4 两个函数配合 Bird_update 函数可以实现通过持续按动“↑”“↓”按键控制鸟的位置。

```
90  void Bird::Bird_update(){
91      time++;
92      speed+=0.1;
93      switch(nDirection){
94      case 1:
95          vBird.setTop(vBird.top()-20);
96          vBird.setBottom(vBird.bottom()-20);
97          break;
98      case 2:
99          vBird.setTop(vBird.top()+20);
100         vBird.setBottom(vBird.bottom()+20);
101         break;
102     default:;
103     }
104
105     for(int i=0;i<4;i++){
106         barrier[i].up.setLeft(barrier[i].up.left()-(15+speed));
107         barrier[i].up.setRight(barrier[i].up.right()-(15+speed));
108         barrier[i].down.setLeft(barrier[i].down.left()-(15+speed));
109         barrier[i].down.setRight(barrier[i].down.right()-(15+speed));
110     }
111
112     //判断是否需要新建障碍
113     double judge_bar=15*time+0.05*time*time;
114     if(judge_bar>=bar_num*150){
115         CreateBarrier();
116         bar_num++;
117     }
118
119     update();
120     QString totaljump=QString::number(total_jump);
121     ui->number->setText(totaljump);
122
123 }
```

图 4.5: 连接计时器的更新函数

在图 4.5 函数中，实现了对鸟的位置的改变的更新、实现随着游戏事件的增长障碍物移动速度的增加以及判断是否需要刷新障碍物。

4.2 游戏进程相关实现

```
125 void Bird::BirdJudge(){
126     //判断游戏是否结束
127     for(int i=0;i<4;i++){
128         if((vBird.right()>barrier[i].up.left())&&(vBird.left()<barrier[i].up.right())){
129             if((vBird.top()<barrier[i].up.bottom())||(vBird.bottom()>barrier[i].down.top())){
130                 blsOver=true;
131 
132                 break;
133                 update();
134                 return;
135             }
136         }
137     }
138 
139     //判断是否越过新的障碍
140     for(int i=0;i<4;i++){
141         if((vBird.right()>barrier[i].up.left())&&(vBird.left()<barrier[i].up.right())){
142             if((vBird.top()>barrier[i].up.bottom())&&(vBird.bottom()<barrier[i].down.top())){
143                 if(i==jump_n){
144                     break;
145                 }
146                 jump_n=i;
147                 total_jump++;
148                 break;
149             }
150         }
151     }
152 
153 
154     update();//paintEvent更新
155 }
```

图 4.6: 判断游戏是否结束的函数

图 4.6 函数通过比较鸟和障碍物的坐标来判断鸟的位置，如果鸟撞到障碍物，则游戏结束；若鸟越过障碍物，则越过障碍物数量增加。

```
157 //创建障碍
158 void Bird::CreateBarrier(){
159     int BarrierUp;
160     BarrierUp=2+qrand()%4;
161     int BarrierLength;
162     BarrierLength=2+qrand()%2;
163     int m;
164     m=BarrierUp+BarrierLength;
165
166     QRect rectup(440,0,40,BarrierUp*40);
167     QRect rectdown(440,m*40,40,(10-m)*40);
168     barrier[barriernum%4].up=rectup;
169     barrier[barriernum%4].down=rectdown;
170     barriernum++;
171
172     update();
173 }
```

图 4.7: 暂停/继续的函数

图 4.7 函数通过暂停计时器实现游戏的暂停和继续。

```
176 void Bird::on_stopBtn_clicked()
177 {
178     QString button_style="QPushButton{background-color:#4682B4;color:white}";
179     QString button_style2="QPushButton:pressed{backgroung-color:white;color:#4682B4;border-style:inset;}";
180     if(blsStop==false){
181         ui->stopBtn->setText(QString::fromUtf8("继续"));
182         ui->stopBtn->setStyleSheet(button_style2);
183         timer1->stop();
184         blsStop=true;
185     }
186     else{
187         ui->stopBtn->setText(QString::fromUtf8("暂停"));
188         ui->stopBtn->setStyleSheet(button_style);
189         timer1->start(100);
190         blsStop=false;
191     }
192 }
```

图 4.8: 创建障碍物的函数

图 4.8 所示函数将障碍物的坐标存在了一个类的数组中，每次更新覆盖之前的坐标，减少所需内存。

```
194 void Bird::on_pushButton_clicked()
195 {
196     Bird* newBird=new Bird();
197     this->close();
198     newBird->show();
199 }
```

图 4.9: 重新开始游戏

如图 4.9 所示，通过设计师界面按钮连接槽函数实现重新开始游戏。

5 单元测试

图 5.10: 游戏开始界面

图 5.11: 游戏介绍界面

图 5.12: 游戏进行中界面

图 5.13: 游戏暂停界面

图 5.14: 游戏结束界面

6 收获

1.QT 基本功能的使用

学会了一些 QT 的函数的使用。比如计时器、ui 设计及槽函数等。

2. 界面跳转

学会了通过动态分配窗体类的指针实现新窗体的展示。