贪吃蛇游戏设计文档

文档最后更新日期

2020.08.30

概述

本次大作业内容主要是通过使用 Qt 开发出一款贪吃蛇小游戏，整个程序即通过 Qt 官方提供的各种类与框架上实现。

程序功能介绍

主体游戏界面为 40*40 的网格，此外还包括菜单栏、工具栏和按钮，三者均可以实现开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏和载入游戏的功能。

整个游戏可以划分为四个状态：

未开始状态

首次打开游戏，用户处于未开始的状态，游戏界面上只存在长度为 2 的初始贪吃蛇，此时，用户通过鼠标点击空白网格，可在点击处设置障碍，而再次点击障碍可将其重置为空白网格。之后可点击开始游戏按钮即可开始

除此之外。用户还可点击载入游戏，选择文件来载入一个已保存的格局，之后游戏进入暂停状态。

游戏状态：

点击开始游戏后，游戏进入游戏状态：程序会随机选择一个空白网格生成一个果实。此后，按任意方向键，贪吃蛇将向对应方向移动，并开始计步，在游戏过程中，程序可实时显示出贪吃蛇头部移动的格子数。而当贪吃蛇吃到果实后，在接下来的 3 步中，贪吃蛇的头部会延长出 3 个格子，与此同时尾部位置保持不变。

此外用户通过点击暂停按钮来暂停游戏。

暂停状态：

点击暂停游戏后，游戏会进入暂停状态，贪吃蛇的移动停止，计步器也随之停止。

此时点击保存游戏，可以保存游戏当前的格局。由用户指定保存文件的名称

与位置。

点击继续游戏后，贪吃蛇恢复移动，恢复计步，再次进入游戏状态。

终止状态：

当满足失败条件时，游戏会进入终止状态，提示游戏失败，游戏计步停止，贪吃蛇无法移动。

游戏的失败条件为：

1. 贪吃蛇碰到周围边界

2. 碰到障碍

3. 碰到自己的身体

当处于暂停状态或终止状态时，点击重新开始，用户变为未开始状态，计时清零，格局恢复为初始状态

游戏处于任何状态时，点击退出游戏，游戏直接关闭。

在特定的状态下，用户只可点击操作页面上的部分按钮，具体的规则如下

未开始状态：暂停游戏、继续游戏、重新开始、保存游戏处于不可用状态，开始游戏、退出游戏、载入游戏处于可用状态。

游戏状态：开始游戏、载入游戏、继续游戏、重新开始、保存游戏处于不可用状态，退出游戏、暂停游戏处于可用状态。

暂停状态：开始游戏、暂停游戏、载入游戏处于不可用状态，退出游戏、继续游戏、重新开始、保存游戏处于可用状态。

终止状态：开始游戏、暂停游戏、继续游戏、载入游戏、保存游戏处于不可用状态，退出游戏、重新开始处于可用状态。

代码设计思路

贪吃蛇游戏的核心实现逻辑

核心逻辑：本程序主要是通过记录贪吃蛇所占的各个网格的位置来存储贪吃蛇的当前位置数据，之后根据用户的下一步输入以及游戏的玩法规则，来相应地改变所存储的贪吃蛇数据，并重新绘制整个画面，以达到控制游戏中小蛇移动、寻找果实、撞击障碍物等动作的效果

1. 贪吃蛇 Snake 类

本次大作业中主要使用自定义 Snake 类来存储游戏中小蛇的相关数据。

Snake 类的声明如下图所示

```
class Snake : public QMainWindow{
    Q_OBJECT
private:
    struct  SnakeNode{  //蛇坐标结构
        int x;
        int y;
    };
    const int Margin=80;
    const int mysize=20;
public:
    explicit Snake(QWidget *parent = nullptr);
    QVector<SnakeNode> snakevec;    //蛇坐标容器
    void Move(char key);    //蛇移动主要函数
    void DeteDirMove(char key, SnakeNode &nexthead);    //根据key值确定蛇的移动方向
    void Reset();//重置蛇
    void setData(QVector<int>& tx,QVector<int>& ty);
    QVector<int> snakex();
    QVector<int> snakey();
};
```

为方便存储位置数据,作业中在 Snake 类中定义了一个私有类型 SnakeNode，其中包含了一个结点的横纵坐标位置信息，之后定义了一个此类型的 QVector，用以存储整个贪吃蛇的各个结点。这样一来，当控制 Snake 的移动时，只需要在 Move 函数中依据一定规则修改 snakecvec 容器中的结点即可。

2. 果实 Food 类

```
class Food : public QMainWindow{
    Q_OBJECT
private:
    int foodx = 0;     //食物坐标
    int foody = 0;
    const int Margin=80;
    const int mysize=20;
public:
    explicit Food(QWidget *parent = nullptr);
    void randfood(QVector<int> sx,QVector<int> sy,bool isWall[][40]);    //随机产生食物坐标
    int getfoodX()const;     //返回食物坐标
    int getfoodY()const;
    void setfoodPos(int _x,int _y);
};
```

Food 类中主要数据成员为 foodx 和 foody，两个整数分别记录了 food 的横纵坐标以供程序可以在相应的位置绘制出果实。当游戏中贪吃蛇吃掉果实后，类中 randfood 函数会参照蛇以及障碍物的位置在空白网格处随机生成一个新的坐标，并赋值给当前对象的两个成员变量。

3. 墙的绘制

由于游戏的界面是 40*40 的网格，因此，直接在 mainwindow 类中使用了一二维数组来记录游戏的空白网格的状态。在 mainwindow 类中重写了其

pressevent 事件函数，使得在游戏处于未开始的状态时，点击游戏中的空白网格，会修改点中的网格状态，相关的实现如下

```
if(gamestart) return;
if(event->button()!=Qt::LeftButton){
    return;
}
int x=(event->x()-Margin)/mysize-1;
int y=(event->y()-Margin)/mysize-1;
if(x>=0&&x<40&&y>=0&&y<40){
    isWall[x][y]^=true;
    update();
}
return;
```

调用 event 的相关函数获取点击位置，并在 iswall 中将相应的数据修改为相反值，最后依据 iswall 数组所存储的各个网格的状态来绘制游戏背景界面。其中的 gamestart 用以判断游戏是否已经开始，若已开始，则下面的代码无效，不可再更改游戏中障碍物的数量及位置。

4.贪吃蛇吃到果实的判断以及之后的奖励机制

在 mainwindow 类中创建一个计时器，计时器可以一定时间间隔不断的发出信号以定时调用信号所绑定的函数，在定时器所绑定的函数中，调用 snake 对象的 move 函数，使得小蛇产生移动，同时，每次移动后判断小蛇头部位置与 food 对象的位置是否重合，显然如若重合，则代表小蛇此步吃到果实，触发奖励机制，若没有吃到，则将 snake 对象中 snakevec 容器的尾结点删去，表示小蛇只向前移动一步，而总长未发生变化。

```
snake.Move(key);    //蛇移动函数
dist++;
//吃到食物的情况
if(reward>0||(food.getfoodX() == snake.snakevec.at(0).x && food.getfoodY() == snake.snakevec.at(0).y)){
    //随机产生一个食物坐标
    if((food.getfoodX() == snake.snakevec.at(0).x && food.getfoodY() == snake.snakevec.at(0).y)){
        food.randfood(snake.snakex(),snake.snakey(),isWall);
        if(reward==0){
            snake.snakevec.pop_back();
            reward+=3;
            ui->label->setText(QString::number(dist));
            update();
            return;
        }
        reward+=3;
    }
    if(reward>0){
        reward--;
    }
}
else{
    snake.snakevec.pop_back();   //删除蛇尾
}
GameOver();    //游戏失败的情况
update();      //调用绘图函数
//更新标签控件的文本内容
ui->label->setText(QString::number(dist));
return;
```

由于作业要求吃到果实后需要在之后的 3 步中小蛇的头部增长 3 个结点，尾部的位置保持不变。因此，本作业中使用一 reward 整型数据来记录小蛇头部需要延长的格子数，如果它大于 0，则不需要删除尾结点，只需头部向对应的方向插入（移动）一个即可，整个功能具体的实现如上图所示。

5. 游戏的终止判断

与奖励机制类似，游戏的终止条件判断也需要和计时器 timer 绑定在一起，每当小蛇移动一步后，都需判断是否满足游戏终止条件。根据题目要求，终止条件有三种，分别是小蛇撞击到外围墙壁，自己以及障碍物，具体的实现代码如下：

```
//撞墙失败
if(snake.snakevec.at(0).x >= 41 * mysize+Margin || snake.snakevec.at(0).x <= Margin ||
   snake.snakevec.at(0).y <= Margin || snake.snakevec.at(0).y >= 41 * mysize+Margin){
    gameTermination();
    return;
}

//撞自己失败
for(int i = 1; i < snake.snakevec.size(); i++){
    if(snake.snakevec.at(0).x == snake.snakevec.at(i).x &&
       snake.snakevec.at(0).y == snake.snakevec.at(i).y){
        gameTermination();
        return;
    }
}
//撞着障碍物失败
//可以这样，遍历所有，如果有的地方的节点的值是true的话，就失败了
for(int i = 0;i<snake.snakevec.size();++i){
    int x=(snake.snakevec[i].x-Margin)/mysize-1;
    int y=(snake.snakevec[i].y-Margin)/mysize-1;
    if(isWall[x][y]){
        gameTermination();
        return;
    }
```

6. 游戏界面的绘制

重写了 mainwindow 类的 paintevent 函数，在此函数中可根据 snake，food 对象的数据成员以及 iswall 数组的值来绘制出先关的小蛇，食物以及障碍物。在每次更改了以上的数据后，根据 Qt 的机制，都可通过调用 update 函数来自动调用 paintevent 函数进行界面的重绘，依此来及时显示出小蛇、食物以及障碍物的图像。

程序的其他控制功能实现

针对程序中所含的按钮，菜单栏选项以及工具栏，本程序主要采用 Qt 中的信号与槽机制实现，分别将上述种种控件所发出的信号绑定到相关的函数上，当用户点击控件发出信号后，会自动调用相关的函数以实现对游戏的控制。

根据作业要求，游戏的状态可分为四种，分别是未开始状态，游戏状态，暂停状态和终止状态，因此本程序将四种状态都写为了函数，以便按钮与点击后要进入的状态进行绑定。

```
void MainWindow::notStarted(){
    reward=0;
    snake.Reset();      //重置蛇
    dist = 0;          //得分清0
    key = 'd';          //key值向右
    timer->stop();//重置定时器
    gameflag = false;   //重置游戏结束标志
    for(int i=0;i<40;++i)
        for(int j=0;j<40;++j)
           isWall[i][j]=false;
    gamestart=false;    //游戏是否是未开始状态
    bugflag=false;
    food.setfoodPos(0,0);
    ui->label->setText(QString::number(dist));

    this->ui->btnLoad->setEnabled(true); this->ui->load->setEnabled(true);
    this->ui->btnStart->setEnabled(true); this->ui->start->setEnabled(true);
    this->ui->btnQuit->setEnabled(true); this->ui->quit->setEnabled(true);
    this->ui->btnPause->setEnabled(false); this->ui->pause->setEnabled(false);
    this->ui->btnCon->setEnabled(false); this->ui->con->setEnabled(false);
    this->ui->btnRestart->setEnabled(false); this->ui->restart->setEnabled(false);
    this->ui->btnSave->setEnabled(false); this->ui->save->setEnabled(false);
}
void MainWindow::gameStarted(){

    timer->start(timerspeed);
    gamestart=true;

    this->ui->btnLoad->setEnabled(false);  this->ui->load->setEnabled(false);
    this->ui->btnStart->setEnabled(false); this->ui->start->setEnabled(false);
    this->ui->btnQuit->setEnabled(true);   this->ui->quit->setEnabled(true);
    this->ui->btnPause->setEnabled(true);  this->ui->pause->setEnabled(true);
    this->ui->btnCon->setEnabled(false);   this->ui->con->setEnabled(false);
    this->ui->btnRestart->setEnabled(false);this->ui->restart->setEnabled(false);
    this->ui->btnSave->setEnabled(false);  this->ui->save->setEnabled(false);
}

void MainWindow::gamePaused(){
    timer->stop();
    ui->label->setText(QString::number(dist));

    this->ui->btnLoad->setEnabled(false); this->ui->load->setEnabled(false);
    this->ui->btnStart->setEnabled(false); this->ui->start->setEnabled(false);
    this->ui->btnQuit->setEnabled(true);   this->ui->quit->setEnabled(true);
    this->ui->btnPause->setEnabled(false); this->ui->pause->setEnabled(false);
    this->ui->btnCon->setEnabled(true);    this->ui->con->setEnabled(true);
    this->ui->btnRestart->setEnabled(true);this->ui->restart->setEnabled(true);
    this->ui->btnSave->setEnabled(true);   this->ui->save->setEnabled(true);
}
void MainWindow::gameTermination(){
    gameflag=true;
    timer->stop();  //暂停定时器

    this->ui->btnLoad->setEnabled(false);  this->ui->load->setEnabled(false);
    this->ui->btnStart->setEnabled(false); this->ui->start->setEnabled(false);
    this->ui->btnQuit->setEnabled(true);   this->ui->quit->setEnabled(true);
    this->ui->btnPause->setEnabled(false); this->ui->pause->setEnabled(false);
    this->ui->btnCon->setEnabled(false);   this->ui->con->setEnabled(false);
    this->ui->btnRestart->setEnabled(true);this->ui->restart->setEnabled(true);
    this->ui->btnSave->setEnabled(false);  this->ui->save->setEnabled(false);
}
```

具体的代码如上所示，因实现逻辑较为简单，只需更改相应的按钮属性以及 timer 的状态等即可，故不再赘述

退出功能实现

将退出按钮或菜单项直接绑定到 mainwindow 的 close()函数去，即可实现在任意状态下点击便可直接退出。

游戏格局的保存与载入

保存

在用户点击保存按钮并指定保存路径后，程序会根据所指定路径创建一个 mdat 文件，在 mdat 文件中，程序会以字符串的形式将食物的位置信息，障碍物的位置信息，小蛇的当前位置信息以及前进方向，当前的总步数等写入，具体代码如下所示

```
QString path=QFileDialog::getSaveFileName(this,"save","../","MDAT(*.mdat)");
if(path.isEmpty()==false){
    QFile file(path);
    bool isOK=file.open(QIODevice::WriteOnly);
    if(isOK){
        QString str;
        str=str+"FOOD"+"\n"+QString::number(food.getfoodX())+"\n"+QString::number(food.getfoodY())+"\n";
        str+="ISWALL\n";
        QString temp;
        for(int i=0;i<40;++i){
            for(int j=0;j<40;++j){
                if(isWall[i][j]){
                    temp+=(QString::number(i)+" "+QString::number(j)+"\n");
                }
            }
        }
        QVector<int> tempx(snake.snakex()),tempy(snake.snakey());
        QString temp1,temp2;
        for(auto x:tempx)
            temp1+=(QString::number(x)+" ");
        for(auto y:tempy)
            temp2+=(QString::number(y)+" ");
        temp1.remove(temp1.size()-1,1);
        temp2.remove(temp2.size()-1,1);
        str=str+temp+"SNAKE\n"+temp1+"\n"+temp2+"\n"+ui->label->text()+"\n"+QString::number(reward)+"\n"+key;
        file.write(str.toUtf8());
    }
    file.close();
}
```

读取

根据保存数据时所规定的格式，使用 QFile 类的 readline 函数，将数据以行的方式读入，之后对每行数据进行处理并修改 mainwindow 中相关类成员的值即可。

如读入用户所定义的墙的数据：

```
//读用户自定义的墙的数据
array=file.readLine();
while(1){
    array=file.readLine();
    str=array;
    if(str=="SNAKE\n") break;
    QStringList templist=str.split(" ", QString::SkipEmptyParts);
    int t[2];
    for (int i = 0; i < templist.size(); ++i)
          t[i]=templist.at(i).toInt();
    isWall[t[0]][t[1]]=true;

}
```

读入保存时蛇的当前位置的数据：

```
//读蛇的位置
QVector<int> tx,ty;
array=file.readLine();
str=array;
QStringList tempx=str.split(" ",QString::SkipEmptyParts);
for (int i=0;i<tempx.size();++i) {
    tx.push_back(tempx.at(i).toInt());
}
array=file.readLine();
str=array;
QStringList tempy=str.split(" ",QString::SkipEmptyParts);
for (int i=0;i<tempy.size();++i) {
    ty.push_back(tempy.at(i).toInt());
}
```