黄金矿工小游戏项目报告

一、程序功能介绍

我们组制作了一款模仿黄金矿工的小游戏，实现了以下界面和功能：

开始界面：背景展示了游戏名称和主要角色矿工老爷爷。窗口右侧对游戏道具和游戏规则进行了介绍，例如延长游戏时间的钟表、获取随机数额奖励的袋子。窗口左下侧有 3 个按钮，点击“普通模式”即可进入基础版黄金矿工游戏；点击“惊喜模式”进入特殊道具关卡，该关卡的随机道具全部为神秘袋子；点击“连续关卡”即可进入带有商店界面的连续关卡版黄金矿工游戏。

游戏主界面：矿工爷爷的钩子以某个速率匀速转动，玩家可通过按下键盘上的向下键操纵其向下释放钩子从而抓取场景中的物品。场景中的物品以随机概率出现，有不同大小和形状的金块、各色的宝石、加减速药水，延长游戏时间的钟表、获取随机数额奖励的袋子。连续关卡里我们还实现了一个特色功能——奔跑的小猪。在第三关的场景中会有小猪左右来回奔跑，如果抓住了小猪，就可以获得价值 500 的金钱奖励。主游戏界面上方为工具栏，分别是展示当前金币数目和剩余时间的显示框、刷新按钮、背景音乐开关和暂停按钮。按下刷新按钮，即可刷新当前场景中的物品种类、数目和布局，剩余时间和金币数目也会回到初始状态。按下键盘上的空格键和工具栏里的暂停键均可进入游戏的暂停界面。基础版中当剩余时间为 0 时弹出结束界面；连续关卡版中弹出商店界面或者结束界面。连续关卡通过不断切换显示主界面和商店界面实现，并用一套变量记录需要继承的属性。

暂停界面：该界面的主要部分是规则介绍。界面下方有“继续”按钮，点击该按钮或者按下空格键即可继续刚才的游戏。

商店界面：该界面的背景是一个货架。商店一共有以下 6 种商品：可以加快矿工提升钩子速度的加速药水、延长游戏时间的钟表、提升钩子承重能力的合金缆绳、增加神秘袋子获得更高价值奖励概率的三叶草、增加宝石价值的宝石上光剂和炸掉钩子上物品的炸药。商品的下方是价格标签，点击价格标签会弹出介绍商品信息的信息窗口，点击“确定购买”按钮如果剩余金钱数目足够则会显示“购买成功～”，否则显示“对不起～努力赚更多金币吧”。商店的右侧工具栏有显示当前拥有金币数目的显示框、continue 按钮和 exit 按钮。按下 continue 就会进入下一关卡继续游戏，按下 exit 就会退出游戏进入结束界面。

结束界面：界面中央显示“game over”和游戏中总共获得的金币数目。下方有两个按钮，点击重新开始可以开始一局新游戏，点击退出则能够关闭游戏界面。

二、项目各模块和类设计细节

item.h&item.cpp

```
#ifndef ITEM_H
#define ITEM_H
class Item
{
private:
    int x,y;//该物体的坐标
    int height,width;
    int price;
    int kind;//0表示普通物质，1为加速，2为减速
    double speed;
    bool visible;
public:

    Item(){
        visible=true;
    }
    void set_position(const int &xx,const int &yy);//设置中心位置
    void set_size(const int &wwidth,const int &hheight);
    void set_price(const int &pprice);
    void set_speed(const double &sspeed);
    void set_kind(const int &kkind);
    void set_visible();//设置为可见
    void set_invisible();//设置为不可见

    int get_kind(){return kind;}
    int get_price(){return price;}
    int get_x(){return x;}
    int get_y(){return y;}
    double get_speed(){return speed;}

    bool is_visible(){return visible;}


    void init();

    bool detection(const int &xx,const int &yy);//碰撞检测
};

#endif // ITEM_H
```

item 类实现了场景中的各类物品的基类。该类具有以下属性：坐标 xy、长和宽、价格、速度、可见属性和种类。种类属性的设置是为了减少不必要的派生类使得代码更加精简易懂，事实上场景里大部分物块没有本质上的区别，所以我们可以通过种类来直接判断物品的类型。类的设计我尽量做到了符合代码规范，访问和更改私有属性必须通过相应的函数。item 里最重要的函数是碰撞检测函数——

```
bool Item::detection(const int &xx,const int &yy)//item碰撞检测
{
    if(visible==true&&xx>x+5&&xx<(x+width-5)&&yy>y+2&&yy<(y+height-5))//+5、-5是为了缩小碰撞检测范围
        return true;
    return false;
}
```

实现原理其实比较简单，就是通过检测钩子和物品长和宽之间的重叠范围，当重叠范围长度大于 5 时就认为钩子抓取到了物品。

miner.h&miner.cpp

miner 类主要实现了矿工爷爷的钩子。钩子具有以下私有属性：int 类型的当前旋转角度 angle，double 类型的伸展速度 speed、当前绳子长度 length 和物块从被抓起来的初始位置到当前位置的距离，bool 类型的参数包括记录是否正在旋转的 is_rotating、是否正抓取了物体的 is_grab、是否停止运动的 stopped 和是否需要更新矿工爷爷动作的 fflag。为了方便修改，我们将记录抓取的价值总数的 sum 设置成了公有属性。

```
class Miner
{
private:
    int angle;
    double speed;
    double length;
    double rlength;


    bool is_rotating;
    bool is_grab;
    bool stopped;
    bool fflag;

public:
    //关于各项数值的更改
    int sum;
    void change_angle(const int &aangle);
    void change_length(const double &llength);
    void change_rlength(const double &rrlength);
    void change_sum(const double &ssum);

    //关于各项数值的设置
    void set_length(const double &llength);
    void set_rlength(const double &rrlength);
    void set_speed(const double &sspeed);

    //关于各布尔值的设置
    void set_is_rotating(bool c);
    void set_is_grab(bool c);
    void set_stopped(bool c);

    //get数值以及布尔值
    double get_speed(){return speed;}
    int get_angle(){return angle;}
    double get_length(){return length;}
    double get_rlength(){return rlength;}
    int get_sum(){return sum;}

    bool isrotating(){return is_rotating;}
    bool isgrab(){return is_grab;}
    bool isstopped(){return stopped;}
    bool isfflag()
    {
        fflag=!fflag;
        return fflag;
    }
    //初始化
    void init();

};

#endif // MINER_H
```

```
void Miner::change_angle(const int &aangle)
{
    angle+=aangle;
}
void Miner::change_length(const double &llength)
{
    length+=llength;
}
void Miner::change_rlength(const double &rrlength)
{
    rlength+=rrlength;
}
void Miner::change_sum(const double &ssum)
{
    sum+=ssum;
}
```

鉴于计算机的离散特性，我们通过在每个较短的时间间隔里设置一个很小的改变量来实现用户界面的视觉连续效果。同样的方法也应用在了实现矿工爷爷的拉伸缆绳的动态动作上，我们反复更改 fflag 的值，并在它为真或者假时刷新不同的矿工的姿势图像。（这个功能主要在 mainwindow 里实现）

map.h&map.cpp

map 类主要用于管理游戏主界面场景里的各个物品和抓取事件。map 类具有私有属性 cur_item 用于记录当前钩子上物

品编号。公有属性有记录场景里所有物品的 Item*数组 item 和记录小猪初始 y 坐标的数组 sy。

```
class Map
{
private:

    int cur_item;
public:
    Map();
    Item *item[18];
    int sy[4]={60,140,290,430};
    int get_cur_item(){return cur_item;}//获取当前物体编号
    double get_cur_speed(){return item[cur_item]->get_speed();}//获取当前物体的速度
    int get_cur_price(){return item[cur_item]->get_price();}//获取当前物体的价值
    int get_cur_kind(){return item[cur_item]->get_kind();}
    QPoint get_point(const int &i);//获取第i个物体的坐标(QPoint)
    int get_kind(const int &i);
    bool is_visible(const int &i);//查看i物体是否可见

    bool detection(const int &x,const int &y);//碰撞检测

    void invisible();//使cur_item不可见
    void rrand();//随机生成矿石
    void rrand2();//只生成袋子
    void rrand3();//有石头
    void run_pig(bool dir);//奔跑的猪

    void init();
    void init_pig();
};
#endif // MAP_H
```

为了避免频繁刷新窗口给玩家在视觉效果上造成的不舒适，我们采取了设置 item 的 visible 属性来控制控件在游戏主界面上的显示和隐去。map 的 detection 函数通过遍历场景里所有的物品来检测钩子是否能够抓取到物品。rrand 系列函数用于随机生成场景里的物品，特殊的分段生成方式在具有随机性的同时也保证了生成物品的合理性。以下是一段实例——

```
void Map::rrand()
{
    srand(time(NULL));
    int times=1+rand()%3;//随机生成1-3个
    int count=0;
    while(count<times)
    {
        int i=6+rand()%4;
        if(item[i]->is_visible()==false)
        {
            item[i]->set_visible();
            int w=50+rand()%600;
            int h=120+rand()%400;
            item[i]->set_position(w,h);
            if(i==6)//如果是大矿石
            {
                item[i]->set_size(140,120);
                item[i]->set_price(500);
                item[i]->set_speed(-2);
                item[i]->set_kind(0);
            }
            else if(i==9)//是钻石
            {
                item[i]->set_size(60,50);
                item[i]->set_price(700);
                item[i]->set_speed(-3);
                item[i]->set_kind(0);
            }
            else{//如果是小矿石
                item[i]->set_size(60,50);
                item[i]->set_price(200);
                item[i]->set_speed(-4);
                item[i]->set_kind(0);
            }
            count++;
        }
        //只要不是所有随机矿石都被占满，就继续进行循环
        int flag=1;
        for(int i=6;i<10;i++)
            if(item[i]->is_visible()==false)
                flag=0;
        if(flag)
            break;
    }
```

其中 run_pig 和 init_pig 是实现关卡中小猪移动的函数，init_pig 设置小猪的初始位置和价值、碰撞体积等基本属性，run_pig 通过 item 类的 set_position 函数改变小猪坐标。这两个函数在 MainWindow 类中与常规的生成、刷新物品的函数一起被调用，通过同时改变 item 的内部逻辑和 ui 中 label 的外部展示来实现小猪移动的效果。

```
void Map::run_pig(bool dir){
    for(int i=14;i<18;i++){
        if(item[i]->is_visible()){
            int tx = item[i]->get_x();
            int ty = item[i]->get_y();
            if(dir==0){
                item[i]->set_position(tx-10,ty+5);
            }
            else{
                item[i]->set_position(tx-10,ty-5);
            }
        }

    }
}
```

```
void Map::init_pig(){
    for(int i=14;i<18;i++){
        item[i]->init();
        item[i]->set_size(60,50);
        item[i]->set_price(200);
        item[i]->set_speed(-4);
    }
    item[14]->set_position(650,60);
    item[15]->set_position(660,140);
    item[16]->set_position(650,290);
    item[17]->set_position(650,430);

}
```

shop.h&shop.cpp

shop 主要实现了游戏中的商店道具功能。shop 类继承了 QWidget 类，新开了一个用户窗口。shop 类的私有属性还有 name 字符串数组用于记录商品名称，Qlist 数组记录商品价格，QLabel 数组记录各个商品的控件指针。公有属性有记录当前金币数目的 mymoney 和记录时间增加量的 addtime。

```
class shop : public QWidget
{
    Q_OBJECT

private:
    Ui::shop *ui;
    QStringList name = {"力量饮料","钟表","合金缆绳","四叶草","宝石上光剂","炸弹"};
    QList<int> price = {100,200,300,400,500,600};
    QLabel* good[7];

public:
    int mymoney;
    int addtime = 0;
    shop(QWidget *parent = nullptr);
    ~shop();
    //void paintEvent(QPaintEvent* event);
private slots:
    void on_pushButton_clicked();
    void handlepurchasebutton();

    void on_pushButton_2_clicked();
    void handlepurchasebutton2();

    void on_pushButton_3_clicked();
    void handlepurchasebutton3();

    void on_pushButton_4_clicked();
    void handlepurchasebutton4();

    void on_pushButton_5_clicked();
    void handlepurchasebutton5();

    void on_pushButton_6_clicked();
    void handlepurchasebutton6();
    void on_pushButton_7_clicked();
    void on_pushButton_8_clicked();

};
```

该界面的控件主要在 Qt designer 里进行。各个道具的功能在程序功能介绍中已经有所提及。为了将物品规则的摆放在货架背景图片上，我们采用了 QGridLayout 网格布局。为了帮助玩家了解各个道具的功能，我们用 QPushButton 制作了价格标签，点击即可运行槽函数触发相应事件，事件包括对主界面需要修改的参数的记录和物品控件的隐去。在编写过程中，为了体现程序的合理性，我们不仅设置了当前金币数目显示框，还判断了当前金币数目和价格的大小关系来决定是否能成功购买——

```
void MainWindow::on_pushButton_clicked()
{
    QWidget *new_win = new QWidget();
    new_win->resize(200,200);
    new_win->setWindowTitle("力量饮料");

    QPixmap bk(":/shop_image/background.jpeg");
    bk.scaled(new_win->size());
    QPalette palette;
    palette.setBrush(new_win->backgroundRole(),QBrush(bk));
    new_win->setPalette(palette);
    new_win->setAutoFillBackground(true);

    QLabel* label=new QLabel("可以加快下一关中矿工提升的速度\n一次性使用，购买即生效。");
    QPushButton* button=new QPushButton(new_win);
    button->setText("确定购买");
    QVBoxLayout* layout=new QVBoxLayout();
    layout->addWidget(label);
    layout->addWidget(button);
    new_win->setLayout(layout);
    new_win->show();
    QObject::connect(button, &QPushButton::clicked, this, [new_win,this](){
        new_win->close();
        if(sum_of_money>=100){
            handlepurchasebutton();
            MainWindow::ui->label->setVisible(false);
            MainWindow::ui->pushButton->setVisible(false);
        }
        else{
            lack_money();
        }
    });
}
```

shop 类主要实现六种商品，有不同的功能：力量饮料，可以加快下一关中矿工提升的速度，通过将 MainWindow 类中表示提升速度倍率的 k 值从 1 增大为 2 实现；时钟，可以延长下一关的时间限制，通过增大 MainWindow 类中的 time_left 值实现；合金缆绳，可以在下一关中提起更重的物品，map 类中的布尔值 canbelifted 用来标志重物体能否可以被提起，购买合金缆绳后 canbelifted 会从 0 变为 1；三叶草，用来提高神秘袋子的奖励价值，通过修改 map 类中代表袋子基础价值的 dbag 数值实现；宝石上光剂，使下一关中钻石的价值更高，通过修改 map 类中代表钻石价值的 dvalue 数值实现；炸药，炸掉钩子上的物品并收回钩子，实现方法是使当前钩子上的物品所对应 label 不显示、将 miner 类的 is_grab

属性置 0 并将钩子速度 speed 置为正常回缩速度-4。

mainwindow.h&mainwindow.cpp

mainwindow 为主窗口类，实现了游戏的主要功能。

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include<QtGui>
#include<QLabel>
#include<QTimer>
#include<QKeyEvent>

#include<item.h>
#include<miner.h>
#include<map.h>
#include <shop.h>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

private:

    Ui::MainWindow *ui;

    Miner *miner;
    Map *map;
    QLabel *lable[19];//前2个为大金矿，后4个为小金矿,6~9为随机产生的石头（可重复产生）

    QPixmap pix;//钩子
    QImage background,hook;

    int cur_item;
    double rad;//表示1弧度=多少度
    int angle_change;//1：1°  -1：-1°
    double k;//表示倍数（加减速）

    //计时器
    QTimer timer_hook;//钩子旋转伸长缩短刷新（15ms）
    QTimer timer_clock;//倒计时（1s）
    QTimer timer_pig;//倒计时（0.2s）
    QTimer timer_miner_up_down;//矿工向上、向下交替运动（200ms）
    int interval_hook,interval_miner;//15，200
    int time_left;//剩余时间
    int time;//表示已经进行了多久，每8秒随机产生一次矿石
    int time_up,time_down;//表示加速和减速的倒计时
    int mode = 0;
    int countshop = 0;
    shop * shopwindow;
```

```
public:

    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    void paintEvent(QPaintEvent *);
    void keyPressEvent(QKeyEvent *);

    void init();//游戏初始化
    int detection();//碰撞检测， 判断是否碰到物体
    //bool detection();
    void do_stop();//暂停
    void resume();//重新开始
    void updatee();

    int c = 0;


private slots:

    void time_reduce();//倒计时（1s刷新）
    void pig_move();//小猪奔跑
    void rotate();//钩子旋转（15ms刷新）
    void miner_up_down();//矿工向上or向下（200s刷新）
    void hook_length();//绳子伸长or缩短操作(15ms刷新)

    void on_toolButton_start_clicked();//（初始界面）“开始”

    void on_toolButton_reset_clicked();//（游戏界面左上角）重开一局

    void on_toolButton_break_clicked();//（游戏界面）“||”暂停游戏

    void on_toolButton_continue_clicked();//（暂停界面）“继续”

    void on_toolButton_restart_clicked();//（结束界面）“重新开始”

    void on_toolButton_exit_clicked();//（结束界面）“退出”

    void on_toolButton_start2_clicked();
    void on_toolButton_start3_clicked();
    void on_toolButton_ready_clicked();
};

#endif // MAINWINDOW_H
```

上左图通过变量设置了游戏的基本属性，上右图通过函数实现了游戏的主要功能。

下面将详细介绍每个函数的功能、所用到的重要变量、实现方法等。

1. **paintEvent:**用于画钩子和绳，**在后续函数中每当 miner 类中绳子的 angle、length、旋转伸缩状态改变，就会执行 update 函数，重新画图，更新界面状态(避免了在 paintEvent 中加入 update 无限循环的情况)**

根据 miner 的 angle 和 length 属性，需将坐标原点平移至钩子出发点附近，将坐标系旋转一定角度(angle)，再将坐标原点移回原位，再根据 length 画出绳和钩子

```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    painter.translate(350,100); //让图片的中心作为旋转的中心
    painter.rotate(miner->get_angle()); //顺时针旋转90度
    painter.translate(-350,-100); //使原点复原

    int length=miner->get_length();
    if(miner->isrotating()==false||miner->isstopped())
    {
        QPen pen;
        pen.setWidth(6);
        painter.setPen(pen);
        painter.setRenderHint(QPainter::Antialiasing, true);
        painter.drawLine(350,95,350,95+length);
        painter.drawPixmap(320,95+length,pix);//绘图的起点
    }
    else
        painter.drawPixmap(320,95,pix);//绘图的起点
    //update();
}
```

2. **keyPressEvent:**用于接收键盘信息，钩子旋转时**按↓键可下放钩子，按空格键可暂停**、开始游戏，持有炸药、钩子上有物品时，可以通过**按 0 键使用炸药收回钩子**。

```
void MainWindow::keyPressEvent(QKeyEvent *e)
{
    int key=e->key();
    if (!miner->isstopped()&&key ==Qt::Key_Down) // 按向下方向键时
    {
        if(miner->isrotating())//如果正在旋转
        {
            miner->set_is_rotating(false);  //停止旋转，进行下拉操作
        }
    }
    if(key==Qt::Key_Space)//暂停功能
    {
        if(miner->isstopped()&&time_left>0){
            ui->label_rule->setVisible(false);//暂停图案
            ui->toolButton_continue->setVisible(false);
            resume();
        }
        else
            do_stop();
    }
    if(key==Qt::Key_0){
        if(shopwindow->isbought[5]==1&&miner->isgrab()){
            map->invisible();
            lable[map->get_cur_item()]->setVisible(false);

            miner->set_speed(-4);
            miner->set_is_grab(0);

        }
    }
}
```

3. init：游戏初始化，用于游戏开始、重新开始、下一关卡

让剩余时间、金币初始化，工具栏显示初始化，让矿工 miner 状态初始化(钩子伸缩速度置为 1，矿工图片置为初始状态)，让地图及矿石状态初始化(开始时 0-5 矿石可见，6-13 随机矿石不可见)，让控制时间、矿工上下运动、钩子旋转伸缩、小猪运动的计时器开始计时。

```
void MainWindow::init()//游戏初始化
{   k = 1;
    miner->init();//矿工初始化设置
    map->init();//地图初始化设置
    if(mode == 2 && countshop == 2){
        map->init_pig();
        for(int i=14;i<18;i++){
            lable[i]->setVisible(true);
        }
    }

    time_left=60;//剩余时间为60s
    time=time_up=time_down=0;
    ui->lcdNumber_time->display(time_left);
    ui->lcdNumber_sum->display(miner->get_sum());
    if(mode == 2&& countshop == 1){
        lable[7] = ui->label_s1;
    }

    for(int i=0;i<=5;++i)
    {
        lable[i]->setVisible(true);//游戏开始时0-5大小矿石可见
        lable[i]->move(map->get_point(i));
        if(i<2&&mode==0){
            lable[i]->resize(140,120);
        }else{
            lable[i]->resize(60,50);
        }
    }
    for(int i=6;i<=13;++i){  //让随机矿石不可见
        lable[i]->setVisible(false);
    }

    ui->toolButton->setStyleSheet("#toolButton{border-image:url(:/images/iii/miner_init.png);}");//设置矿工

    timer_hook.start();
    timer_clock.start();
    timer_pig.start();
    timer_miner_up_down.start();
}
```

4. do_stop:暂停游戏，用于游戏中点击 break 键暂停游戏

让计时器停止计时，设置矿工钩子旋转伸缩状态

```
void MainWindow::do_stop()
{
    miner->set_is_rotating(false);
    miner->set_stopped(true);
    timer_clock.stop();
    timer_pig.stop();
    timer_hook.stop();
    timer_miner_up_down.stop();
}
```

```
void MainWindow::resume()
{
    miner->set_is_rotating(true);
    miner->set_stopped(false);
    timer_clock.start();
    timer_hook.start();
    timer_pig.start();
    timer_miner_up_down.start();
}
```

5. **resume()**:重新开始，与 do_stop()完全相反，用于暂停后继续游戏

让计时器开始计时，设置矿工钩子旋转伸缩状态

6. **detection()**:检测钩子是否碰撞到物体(如果碰撞到物体则将 miner 的 is_grab 变量置为 true)、是否超出界面左下右边框，返回 1 代表越界、返回 2 代表碰撞、返回 0 代表未发生越界或碰撞，用于下放钩子时判断钩子何时需要伸长缩短，**将越界和碰撞的检测合并为一个函数**，只进行一次判断，**通过返回值来区分不同状态，使代码更简洁**

```
int MainWindow::detection()//碰撞和越界检测     1:越界    2:碰撞
{
    //定义临时变量angle和length
    int angle=miner->get_angle();
    int length=miner->get_length();

    if(fabs(angle)>45){//从左右越界
        if(length>(350/(sin(fabs(angle/rad)))))
            return 1;
    }
    else if(length>(500/(cos(angle/rad))))//从下越界
        return 1;
    if(map->detection(350-length*sin(angle/rad),95+length*cos(angle/rad)))//如果map.碰撞检测返回为真
    {
        miner->set_is_grab(true);
        return 2;
    }
    return 0;
}
```

7. **通过计时器触发的函数**：

(1) **time_reduce()**:由 timer_clock 控制，每 1s 触发一次

若还有时间剩余，则更新游戏进行时间、游戏剩余时间，在加减速药水发挥作用时更新药水发挥作用的剩余时间，**当药水剩余作用时间为 0 时，将绳子伸缩速度置为初始状态 1。每 8s 调用 map->rrand()随机生成矿石一次，并更新界面时间及矿石显示状态。**随模式的不同（普通模式、惊喜模式、随机关卡模式），**调用 map 类中不同的 rrand 函数，随机生成不同的物体**。

若游戏时间结束，则停止游戏。

```
void MainWindow::time_reduce()//计时器  k:倍数  1s
{
    if(time_left>=0)
    {
        ++time;--time_left;
        ui->lcdNumber_time->display(time_left);
        if(time_down==0&&time_up==0)
        {
            k=1;
            //timer_miner_up_down.setInterval(interval_miner);
            //timer_hook.setInterval(interval_hook);
        }
        if(time_down)
        {
            --time_down;
        }
        if(time_up)
        {
            --time_up;
        }
        if(time%8==0)//每8秒，随机生成矿石一次
        {
            if(mode == 0) {
                map->rrand();
            }
            else if(mode == 1) map->rrand2();
            else if(mode == 2){
                if(countshop == 0) map->rrand();
                if(countshop == 1) map->rrand3();
//                if(countshop == 2){

//                }
            }


            for(int i=6;i<=13;++i)
                if(map->is_visible(i))//如果该物体可见
                {
                    lable[i]->setVisible(true);
                    lable[i]->move(map->get_point(i));
                }
        }
    }
    if(time_left<0)//剩余时间为0，则停止游戏
    {

        do_stop();
        if(mode!=2){
        ui->label_finish_background->setVisible(true);//终止背景
        ui->toolButton_exit->setVisible(true);
        ui->toolButton_restart->setVisible(true);
        ui->label_final->setVisible(true);//final: 得分
        ui->toolButton_music->setVisible(false);
        QString s = QString::number(miner->get_sum(), 10);
        ui->label_final->setText(s);
        }
        if(mode==2){


            if(countshop == 0) shopwindow = new shop;
            if(countshop<3){
            shopwindow->mymoney = miner->sum;
            for(int i=0;i<6;i++){
                shopwindow->isbought[i] = 0;
            }
            shopwindow->show();

            countshop++;
            ui->toolButton_ready->setVisible(true);
            }
            else{
                ui->label_finish_background->setVisible(true);//终止背景
                ui->toolButton_exit->setVisible(true);
                ui->toolButton_restart->setVisible(true);
                ui->label_final->setVisible(true);//final: 得分
                QString s = QString::number(miner->get_sum(), 10);
                ui->label_final->setText(s);
            }

        }
    }
    update();
```

(2) rotate():由 timer_hook 控制，每 15ms 触发一次，用于**改变钩子角度**
变量 angle_change 值为 1 或-1，代表钩子每 15ms 角度变化 1°，当钩子角度绝对值>=90°时，将 angle_change 值取反，**可保证钩子在 [-90°， 90°] 内旋转**

```
void MainWindow::rotate()//控制钩子旋转（15ms刷新）  /interval
{
    if(miner->isrotating())//旋转时
    {
        miner->change_angle(angle_change);
        int angle=miner->get_angle();
        if(angle>=90||angle<=-90)
            angle_change*=-1;
    }
    update();
}
```

(3) miner_up_down():由 timer_miner_up_down 控制，每 200ms 触发一次，**用于控制矿工身体上下运动，从而实现动态效果**
miner 的 **isfflag 函数**返回 bool 值，且每调用一次 bool 值取反，从而控制矿工向上、向下两张图片的切换

```
void MainWindow::miner_up_down()//矿工向上or向下（200ms刷新）
{
    if(!miner->isrotating())
    {
        if(miner->isfflag())//矿工的动作设置
            ui->toolButton->setStyleSheet("#toolButton{border-image:url(:/images/iii/miner_down.png);}");
        else
            ui->toolButton->setStyleSheet("#toolButton{border-image:url(:/images/iii/miner_up.png);}");
    }
    update();
}
```

(4) hook_length():由 timer_hook 控制，每 15ms 触发一次，**用于控制绳子伸长缩短、物块被抓、被抓物块随绳子一同运动、钩子把物块抓回起点后物块发挥功效等核心功能**

用 miner 的 isrotating 属性判断钩子是否停止旋转，若为 true，则执行后续操作。

绳子伸缩速度由 miner 的 speed 属性和 k 值控制,每 15ms 绳子长度以 miner->get_speed()*k 的速度变化。
Speed：绳子开始下放时 speed 值为 4，用 detection 函数检测是否越界或碰到物体，若返回值为 1(越界)，则 speed 值置为-4(反向)，若返回值为 2(碰到物体)，则将 speed 根据不同物块置为不同值(大金块:-4，小金块:-2，钻石:-3)
K：由加减速药水控制，加速药水生效时 k 为 3，减速药水生效时 k 为 0.5

当钩子回到起点时(length<=0)，若抓回了物块(isgrab 为 true)，则设置物块不可见，绳子 isgrab 状态为 false，并使抓回的物块发挥作用，先使金币数增加物块对应价值数，**再判断物块 kind 值是否为 0，若不为 0，则说明物块有除金钱增长外的其他作用**。若 kind 值为 1，则发挥加速药水的作用(通过 time_up,k 实现)；若 kind 值为 2，则发挥减速药水的作用(通过 time_down,k 实现)；若为其他情况(时钟)，则增加游戏剩余时间。

最后更新游戏界面(金币数，钩子、绳和被抓物块的状态)

游戏中特别注意了以下功能的实现：
加减速药水不能叠加；
若下放钩子时未抓取到物块，返回时若有新的物块刷新出来，即使钩子与物块相碰撞，也不能抓取物块(钩子只在下放时可以抓取物块，钩子收回时只能将下放时抓取到的物块抓回原点）

```
//speed: miner:4   big:-4   small:-2   diamond:-3
void MainWindow::hook_length()//绳子伸长or缩短操作(15ms)   /interval   /k
{
    if(!miner->isrotating())
    {
        if(miner->isgrab())//物体的移动
        {   //miner::rlength为当前位置到返回起点的距离
            miner->change_rlength(k*fabs(miner->get_speed()));//绳和钩子移动
            int rlength=miner->get_rlength();
            int angle=miner->get_angle();
            QPoint p=map->get_point(map->get_cur_item());//获取当前物体的坐标
            QPoint temp(rlength*sin(angle/rad),-rlength*cos(angle/rad));//temp为位移大小
            lable[map->get_cur_item()]->move((p+temp));//物体移动
        }
        else if(miner->get_speed()>0){
            int result=detection();
            if(result==1){
                miner->set_speed(-1*(miner->get_speed()));
            }else if(result==2){//detection()时已设置miner->isgrab()
                miner->set_speed(map->get_cur_speed());
            }
        }


        miner->change_length(k*miner->get_speed());

        if(miner->get_length()<=0)//如果钩子回到了起点
        {
            if(miner->isgrab())//如果抓到了东西
            {
                map->invisible();//令当前物体不可见
                lable[map->get_cur_item()]->setVisible(false);//令当前物体不可见
                miner->change_sum(map->get_cur_price());
                miner->set_rlength(0);//
                miner->set_is_grab(false);

                if(map->get_cur_kind())//如果抓到了特殊物品
                {
                    int kind=map->get_cur_kind();
                    if(kind==1)//加速
                    {
                        time_up=10;
                        time_down=0;//不能叠加，所以把另一个的时间归零//叠加如何实现？
                        //timer_miner_up_down.setInterval(30);//快速上下
                        //timer_hook.setInterval(interval_hook);//慢上下
                        k=3;
                    }else if(kind==2)//减速
                    {
                        time_down=10;
                        time_up=0;//不能叠加，所以把另一个的时间归零
                        //timer_miner_up_down.setInterval(interval_miner*2);//慢上下
                        //timer_hook.setInterval(interval_hook*2);//慢上下
                        k=0.5;
                    }
                    else//时间
                        time_left+=15;
                }
            }
            miner->set_is_rotating(true);
            miner->set_speed(4);
        }
    }
    ui->lcdNumber_sum->display(miner->get_sum());
    update();
}
```

8. 背景音乐

通过 QMediaPlayer 和 QAudioOutput 类来实现音频播放，通过和按钮连接来控制音乐开关。

```
player = new QMediaPlayer;
audioOutput = new QAudioOutput;
player->setAudioOutput(audioOutput);
connect(player, SIGNAL(positionChanged(qint64)), this, SLOT(positionChanged(qint64)));
player->setSource(QUrl("qrc:/bgm/bgm.mp3"));
audioOutput->setVolume(50);
musicclicked = 0;
```

```
void MainWindow::on_toolButton_music_clicked()
{
        if(musicclicked == 0){
            player->play();
            musicclicked = 1;
        }
        else{
            player->stop();
            musicclicked = 0;
        }

}
```

三、组内分工

杨舒：各个部分代码的整合调试；从基础版到连续关卡的实现；小猪的实现；录制视频；报告书写；
李静雯：用户界面 mainwindow 的实现；报告书写；
熊婧：item、miner、map 和 shop 的实现；报告书写；

四、总结与反思

总结：

小组团结合作，分工明确，互帮互助，无消极怠工情绪。先实现基础版游戏，后增添进阶功能(商店、特殊道具关卡、连续关卡、奔跑的小猪、音乐等)。

反思：

1. 在按 0 键炸掉钩子上的物块时，炸弹爆炸特效未实现
2. private 和 public 有实现，但还不够完善
3. 界面刷新：从最开始在 paintEvent 函数里嵌入 update 函数，改成现在每发生一次变化执行一次 update 函数，是否还有更好的办法?
4. 在设计规划时想法还不够成熟，最开始小组成员分工写了 mainwindow 和 shopwindow 两个主界面，最后合并时遇到了不少麻烦。分工还不够明确，比如三个人都做了美工，每个人的工作比较杂。版本管理方面，最开始 GitHub 掌握得不够熟练，有时候没有在最新版本上修改，熟练掌握 GitHub 后及时改正了该问题。

特别鸣谢：

https://blog.csdn.net/a343902152/article/details/42774191?ops_request_misc=%257B%2522request%255Fid%