基于《滑雪大冒险》的复刻

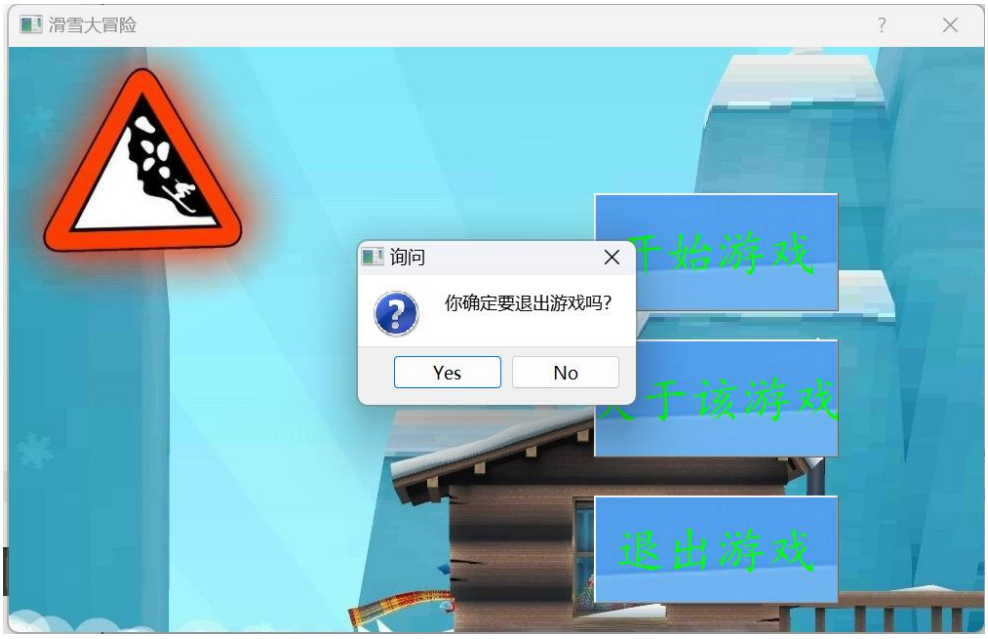
——《程序设计实习》QT 大作业作业报告

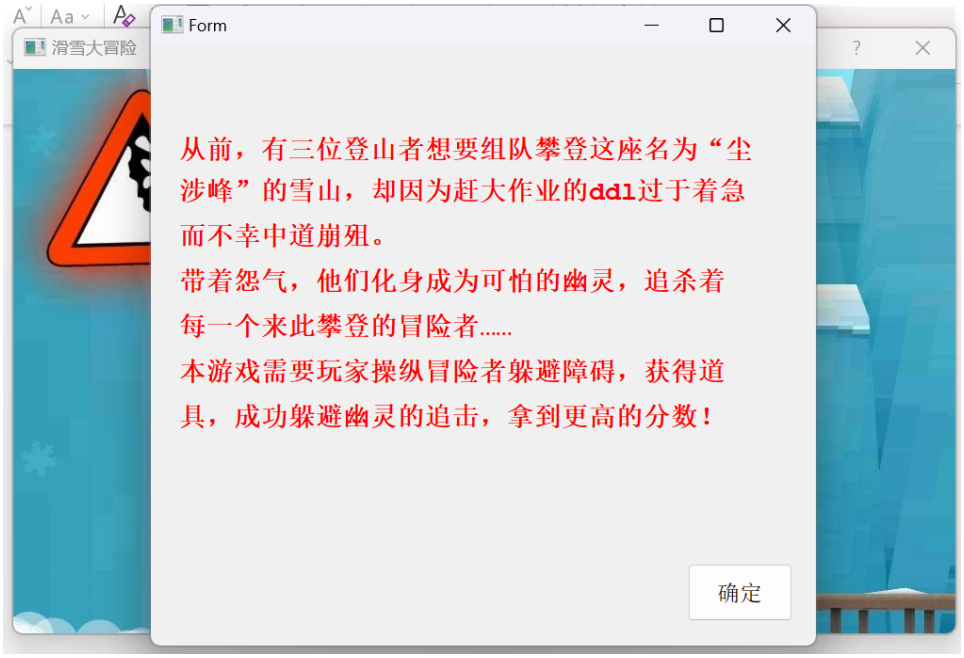
第 37 组 组长：李昀蔚 组员：刘震涛、韦恢航

本小组的 QT 大作业基于游戏《滑雪大冒险》，编写程序对该游戏中的某些功能进行相应的复刻，最终效果为使玩家可以在电脑上进行该游戏的模拟。本 QT 大作业的作业报告分为以下几个方面：①程序功能介绍，②项目各模块与类设计细节，③小组成员分工情况，④项目总结与反思。

一、程序功能介绍

本游戏的加载界面是主菜单，具有“开始游戏”“关于游戏”“退出游戏”三个按钮。点击“退出游戏”时，系统会弹出对话框进行询问；点击“关于游戏”时，会生成文本框，介绍该游戏的背景。





当点击“开始游戏”时，会出现如下图所示的游戏界面：

游戏界面中，“返回”按钮可以让玩家返回至主菜单，“跳跃”按钮可以使在地面（即图中黑线）上的人物进行弹跳。从屏幕右侧会飘出一系列随机的障碍物（石头）和道具（火箭），障碍物会使人物减速，道具会使人物加速。

当鬼魂和人物的距离小于一定值时，图片右上角会显示警示图标，催促玩

家尽快逃离危险。左上角具有与鬼魂距离提示，以及目前总得分（即人物已经前进的水平距离）。

当鬼魂与人物距离为零时，游戏结束，弹出对话框，并且显示玩家的总得分。

二、项目各模块与类设计细节

本部分我们只介绍几个关键性的模块，项目的其他模块可以参看我们组的演示视频。

①主菜单的模块

主菜单中对窗口进行数据设置：

```
setWindowTitle("滑雪大冒险");
setFixedSize(1000,600);
ui->menu_back->resize(1000,600);
picture *mainbg=new picture("C:/Users/lywpc/Desktop/QT/snowpeople/picture/menubackground.jpeg",this);
mainbg->move(0,0);
```

主菜单中建立三个按钮，分别对应相应的槽函数，其中对于“关于该游戏”按钮的槽函数，建立 story.cpp 文件，用来建立相应的故事对话框。

②背景类

本游戏的设计采用相对运动原理，让人物的横坐标不改变，通过背景图片的移动，从视觉上看像是人物向前发生运动。

背景图片可以依据一定的速度发生水平运动。当人物与障碍物或者道具相碰时，背景图片的运动速度会发生调整，从直观上看就相当于人物加速/减速。

```
void BackGround::BackgroundPosition(double addernum)
{
    //处理第一张图片滚动
    m_map1_posX -= BackGround_SCROLL_SPEED-addernum*3;
    if(m_map1_posX <= -GAME_WIDTH)
    {
        m_map1_posX =0;
    }

    //处理第二张图片滚动
    m_map2_posX -= BackGround_SCROLL_SPEED-addernum*3;
    if(m_map2_posX <= 0 )
    {
        m_map2_posX =GAME_WIDTH;
    }
}
```

③地形的建立

由于地形的绘制应该是随机的，我们采用贝塞尔曲线的绘制方法，即取左上角定点和右下角顶点，在其构成的矩形中随机取两个控制点，绘制成一条平滑的曲线。本游戏戏中的地形使用黑色线条绘制。

```
sp1.setX(myx1-0.5*diff1);sp1.setY(myy1-0.3*diff1);
ep1.setX(myx2-0.5*diff1);ep1.setY(myy2-0.3*diff1);//第一段坐标设置

sp2.setX(myx3-0.5*diff2);sp2.setY(myy3-0.3*diff2);
ep2.setX(myx4-0.5*diff2);ep2.setY(myy4-0.3*diff2);//第二段坐标设置

c11.setX((sp1.x()+ep1.x())/2);c11.setY(sp1.y());
c21.setX((sp1.x()+ep1.x())/2);c21.setY(ep1.y());//第一段中间点设置

c12.setX((sp2.x()+ep2.x())/2);c12.setY(sp2.y());
c22.setX((sp2.x()+ep2.x())/2);c22.setY(ep2.y());//第二段中间点设置

QPainterPath path;
path.moveTo(sp1);
path.cubicTo(c11, c21, ep1);
mPainter.drawPath(path);

QPainterPath path2;
path2.moveTo(sp2);
path2.cubicTo(c12,c22,ep2);
mPainter.drawPath(path2);
```

④人物类

人物类具有以下参数：人物的 X 轴方向移动速度（实际上的），人物的 X 坐标，人物的 Y 坐标。为了判断人物是否跳起，加入参数 JumpBool 进行判断。当 JumpBool 的值为 0 时，代表人物未跳起在地面上；值为 1 时，代表人物处于上升状态；值为 2 时，代表人物处于下落状态。

为了符合现实中的弹跳模型，建立了相应的自由落体运动方程。

```
//设重力加速度大小为以下值
m_man_speedY-=0.2;

//当人物落地时，即人物纵坐标不小于地形纵坐标
if (m_man_posY+272>=m_ground_posY&&JumpBool!=1)
{
    m_man_speedY=0;
    JumpBool=0;
}
//当人物开始下落时，修改JumpBool的状态
if (JumpBool==1&&m_man_speedY<=0)
{
    JumpBool=2;
}
```

其中判断地形的纵坐标，我们通过寻找屏幕中的黑像素点进行实现。

```
QPixmap pixmap = list_screen.at(0)->grabWindow(0,x,y,1,1);
if (!pixmap.isNull()) //如果像素图不为NULL
{
    QImage image = pixmap.toImage();//将像素图转换为QImage
    if (!image.isNull()) //如果image不为空
    {
        if (image.valid(0, 0)) //坐标位置有效
        {
            QColor color = image.pixel(0, 0);
            int mousedPressed_R = color.red();
            int mousedPressed_G = color.green();
            int mousedPressed_B = color.blue();
            //QString text = QString("RGB: %1, %2, %3").arg(mousedPressed_R).arg(mousedPressed_G).arg(mouse
            //qDebug() << text;
            //qDebug()<<y<<" "<<mousedPressed_R<<" "<<mousedPressed_G<<" "<<mousedPressed_B<<endl;
            if ((mousedPressed_R==0)&&(mousedPressed_G==0)&&(mousedPressed_B==0))
            {
                StartBlack=y;
                m_people.m_ground_posY=StartBlack;
                //qDebug()<<"初始状态："<<y<<std::endl;
                break;
            }
        }
    }
}
```

⑤道具类、障碍物类

由于这两个类实现原理基本相同，因此我们将道具类和障碍物类设为同一个基类的派生类。人物与障碍物以及道具的碰撞，可通过将人物与障碍物或道具

的 X、Y 坐标进行比较，当两者差值均小于一定值时，视为其相碰，此时对人物的实际速度进行相应的运算。

而道具或障碍物的随机生成，我们采用生成随机数的方法设为道具或障碍物的纵坐标，从屏幕的右侧水平向左运动。为了增加游戏的难度，默认障碍物数量比道具数量多。下图为生成随机障碍物的代码片段：

```
qsrand(QTime::currentTime().msec());
int randnum0;
randnum0=qrand() % 1200;
if (randnum0%100<=3)
{
    m_stone.Stonenum++;

    //依据人物的坐标生成随机数
    int randnum;

        randnum=randnum0/3-100+m_people.m_man_posY;

    m_stone.m_stone_posY[m_stone.Stonenum]=randnum;
    m_stone.m_stone_posX[m_stone.Stonenum]=1000;
}
```

⑥鬼魂类、警示牌类

我们设定鬼魂离人物的距离超过一定值时，警示牌消失，鬼魂移动到屏幕之外。鬼魂和警示牌分别设为两个类。

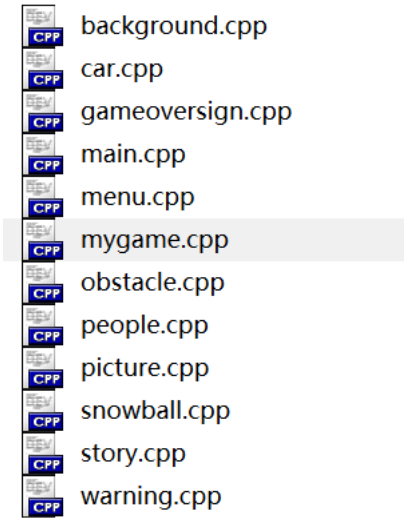
```
void snowball::SnowBallPosition()
{
    const double leftside=-50.0;

    if(m_ball_posX<=leftside&&!FallBool)
    {
        FallBool=1;
    }
    if(m_ball_posX>leftside&&FallBool)
    {
        FallBool=0;
    }
}
```

```
m_warn.load(W_PATH);
m_warn = m_warn.scaled(100, 100, Qt::KeepAspectRatio, Qt::SmoothTransformation);
```

以上的所有类及其成员，均在 mygame.cpp 中进行调用，mygame.cpp 即为游戏主界面运行的程序。

三、小组成员分工情况：



李昀蔚：主菜单（menu.cpp），图片类（picture.cpp），人物类（people.cpp），主程序（main.cpp），作业报告编写，录屏展示。

刘震涛：游戏结束的提示（gameoversign.cpp），鬼魂类（snowball.cpp），游戏背景阐述（story.cpp），警示牌类（warning.cpp）。

韦恢航：背景类（background.cpp），障碍物类（obstacle.cpp），道具类（car.cpp）。

游戏主界面 mygame.cpp 由三人合力完成。

四、项目总结与反思

我们开始绘制地形时，试图寻找一个随机函数，可以用来绘制任意随机形状的曲线，但在查阅相关参考资料时，这样能够确定任意点坐标的函数及其复杂，不是能够简单用 c++程序就能实现的。因此，我们选择了绘制贝塞尔曲线的方法，并将各段绘制的贝塞尔曲线进行收尾相接。

我们组贝塞尔曲线的绘制参考了以下资料：《QT 学习—绘制贝塞尔曲线》https://blog.csdn.net/weixin_42661333/article/details/118391539

但是这么绘制地形有一个很大的缺点：无法确定某一点具体的 X、Y 坐标。因此我们想到了截屏的方法，寻找屏幕中的黑像素点，把黑像素点的 X、Y 坐标设为地形的 X、Y 坐标。实际操作中，由于截屏操作的时间复杂度略高，因此我们在修改参数仍不能实现明显的时间优化后，对程序的逻辑进行了调整，即只在

游戏开始的时候寻找一次黑像素点，随后让该点的坐标进行上下移动，寻找下一个时刻的黑像素点。这么做可以大大减小时间复杂度，但是会造成人物移动时发生轻微的抖动，这个问题我们通过调整帧率使人物抖动变得不明显。

我们组本来设定的道具类、障碍物类均是在地形上生成的，随着地形进行移动，但通过编写程序发现，当地形的 X、Y 坐标同时发生变化时，难以捕捉在时刻黑像素点的位置。因此我们在游戏《滑雪大冒险》的基础上，对其进行了简化，即道具与障碍物是从屏幕右端到左端水平运动的，这样只用编写随机数即可，缩小了道具与障碍物生成的难度。

总体来说，我们小组的此次 QT 大作业，基本实现了复刻《滑雪大冒险》中的各项功能，比如障碍物、道具、积分、人物的滑行与跳跃模型、雪崩（为了趣味性，本大作业中改成了鬼魂，并附加了背景故事）等，但对一些我们认为较为复杂的模型进行了简化。由于我们组的成员均缺乏艺术细胞（笑），因此主菜单即游戏内的界面应该不算很美观，这是我们需要改进的地方。此外，《滑雪大冒险》游戏中的交互功能，本大作业中暂未实现，今后会适当添加一些交互功能。