

记录用户在 Windows 中的操作，并可以重放的软件

目录

- 一、前言3
- 二、开发过程3
 - (一) 设计思路.....3
 - (二) 实现过程.....3
 - (三) 系统功能检测5
- 三、设计方案5
 - (一) 功能组成.....5
 - (二) 界面组成.....5
 - (三) 代码实现.....5
- 四、技术讨论 11
 - (一) 存在的问题 11
 - (二) 解决方案..... 11

一、前言

本次项目是实现一个记录用户在 Windows 中的操作，并可以重放的软件。该项目是一个可以按“录制”后，将用户之后的鼠标移动/点击、键盘输入情况进行记录；点“停止”则停止录制；将上述录制的操作过程记录和保存到文件中；选择一个操作过程，按“重放”可以将记录下来的鼠标移动/点击、键盘输入按原样进行重复执行的软件。

二、开发过程

（一）设计思路

1. 首先先进行软件页面的布局
2. “保存路径”按钮的实现
3. “录制”按钮的实现
4. “停止”按钮的实现
5. “重放”按钮的实现
6. “退出”按钮的实现

（二）实现过程

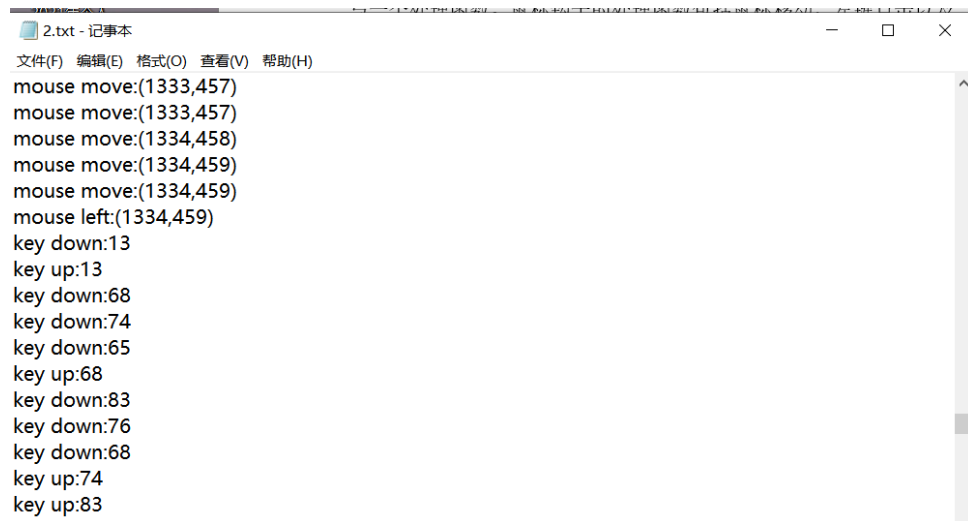
1. 界面设计如下：



2. “保存路径”按钮的实现：
使用 Qt 的文件 `QFileDialog`、`QFile` 实现保存功能。
`fileDialog.getSaveFileName()` 函数实现获取保存文件的路径，并将其保存的路径显示在编辑框上。

3. “录制”按钮的实现

该功能实现的核心技术的钩子(Hook)技术。hook（钩子）是一种特殊的消息处理机制，它可以监视系统或者进程中的各种事件消息，截获发往目标窗口的消息并进行处理。钩子的种类很多，每种钩子可以截获相应的消息，如键盘钩子可以截获键盘消息，外壳钩子可以截取、启动和关闭应用程序的消息等。钩子可以分为线程钩子和系统钩子，线程钩子可以监视指定线程的事件消息，系统钩子监视系统中的所有线程的事件消息。因为系统钩子会影响系统中所有的应用程序，所以钩子函数必须放在独立的动态链接库(DLL)中。hook（钩子）就是一个 Windows 消息的拦截机制，可以拦截单个进程的消息(线程钩子)，也可以拦截所有进程的消息(系统钩子)，也可以对拦截的消息进行自定义的处理。点击录制时创建鼠标钩子和键盘钩子，并对相应的钩子写一个处理函数。创建钩子的函数是 SetWindowsHookEx。鼠标钩子的处理函数包括鼠标移动，左键点击以及右键点击事件的记录，并将其记录放在用户指定文件中。键盘钩子的处理函数包括键盘按下以及抬起，并将过程记录在文件中。记录的文件类似如下：



```
2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
mouse move:(1333,457)
mouse move:(1333,457)
mouse move:(1334,458)
mouse move:(1334,459)
mouse move:(1334,459)
mouse left:(1334,459)
key down:13
key up:13
key down:68
key down:74
key down:65
key up:68
key down:83
key down:76
key down:68
key up:74
key up:83
.
--
```

4. “停止”按钮的实现

点击停止则会卸载鼠标钩子和键盘钩子，即记录结束。卸载钩子的函数是 UnhookWindowsHookEx

5. “重放”按钮的实现

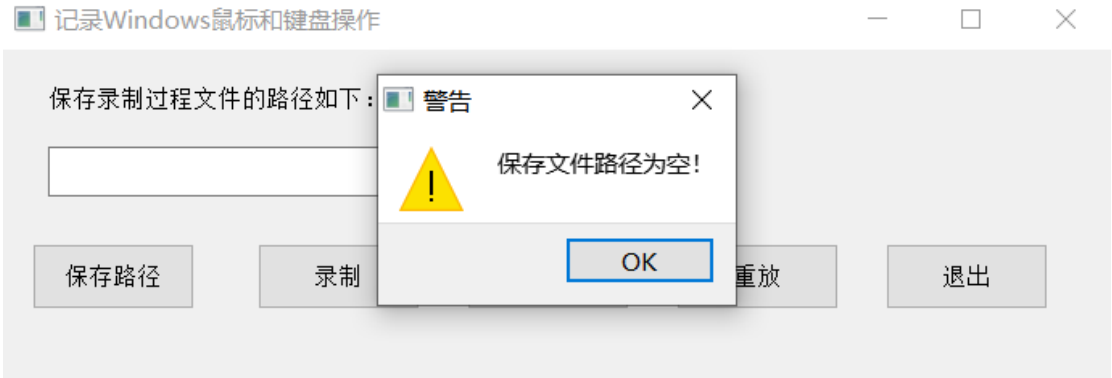
将记录操作的文件打开，调用 windows 鼠标函数 mouse_event 来实现根据文件记录的鼠标移动和点击情况的重现，调用 windows 键盘函数 keybd_event 来实现根据文件的键盘按键情况的重现。

6. “退出”按钮的实现

调用函数 exit (0) 来实现程序的退出。

（三）系统功能检测

1. 检测是否输入了要保存的文件路径，没有的话弹窗警告

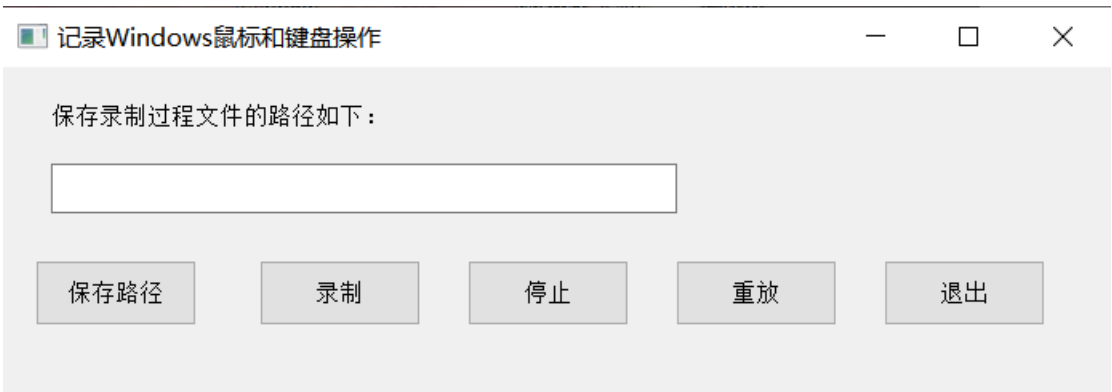


三、设计方案

（一）功能组成

1. 用户选择录制结果保存的文件路径
2. 录制鼠标和键盘的操作
3. 停止录制
4. 重放刚刚录制的结果
5. 退出

（二）界面组成



（三）代码实现

1. 用户选择录制结果保存的文件路径

```

//保存，选择将录制的数据放在哪个文件
connect(ui->pB_save, &QPushButton::clicked, this, [=]() {
    QString curPath=QDir::currentPath();//获取系统当前目录
    QString dlgTitle="请选择存放文件的位置"; //对话框标题
    QString filter="文本文件(*.txt)"; //文件过滤器
    QString
aFileName=QFileDialog::getSaveFileName(this, dlgTitle, curPath, filter);
    if (aFileName.isEmpty())
    {
        QMessageBox::warning(this, "警告", "保存文件路径为空!");
        return ;
    }
    //清空文件
    filename = aFileName;
    QFile tempFile(filename);
    tempFile.open(QIODevice::WriteOnly);
    tempFile.close();
    ui->lineEdit->setText(filename);
    if(filename == NULL)
    {
        QMessageBox::warning(this, "警告", "记录文件路径为空!!!
");
    }
});

```

2. 录制鼠标和键盘的操作

```

void MainWindow::installHook() //创建钩子
{
    keyHook = SetWindowsHookEx(WH_KEYBOARD_LL, keyProc, nullptr, 0);
    //创建键盘钩子
    mouseHook = SetWindowsHookEx(WH_MOUSE_LL, mouseProc, nullptr, 0);
    //创建鼠标钩子
}

//键盘钩子处理函数
LRESULT CALLBACK keyProc(int nCode, WPARAM wParam, LPARAM lParam) //钩
子消息函数，系统消息队列信息会返回到该函数中
{
    if(!keyHook)
        return CallNextHookEx(keyHook, nCode, wParam, lParam);
    if(wParam == WM_KEYDOWN) //wParam 用于判断事件类型，当前为按键
按下事件
    {

```

```

        KBDLLHOOKSTRUCT* pkbhs = (KBDLLHOOKSTRUCT*)lParam; //lParam
用于判断按键类型
        qDebug() << "key down:" << QString::number(pkbhs->vkCode, 10);
        QFile tempFile(filename);
        tempFile.open(QIODevice::Append);
        QString str = "key down:" + QString::number(pkbhs->vkCode, 10)
+ "\n";
        QByteArray strBytes=str.toUtf8();//转换为字节数组
        tempFile.write(strBytes, strBytes.length()); //写入文件
        tempFile.close();
        //可添加一些自定义信号发出去, 如: emit hookClass->keyPress

    }
    else if(wParam == WM_KEYUP)
    {
        KBDLLHOOKSTRUCT* pkbhs = (KBDLLHOOKSTRUCT*)lParam; //lParam
用于判断按键类型
        qDebug() << "key up:" << QString::number(pkbhs->vkCode, 10);
        QFile tempFile(filename);
        tempFile.open(QIODevice::Append);
        QString str = "key up:" + QString::number(pkbhs->vkCode, 10) +
"\n";
        QByteArray strBytes=str.toUtf8();//转换为字节数组
        tempFile.write(strBytes, strBytes.length()); //写入文件
        tempFile.close();
    }
    return CallNextHookEx(keyHook, nCode, wParam, lParam); //继续原
有的事件队列
}

//鼠标钩子处理函数
LRESULT CALLBACK mouseProc(int nCode, WPARAM wParam, LPARAM lParam) //
钩子消息函数, 系统消息队列信息会返回到该函数中
{
    if(!mouseHook)
        return CallNextHookEx(mouseHook, nCode, wParam, lParam);
    //鼠标点击右键
    if (WM_RBUTTONDOWN == wParam)
    {
        MOUSEHOOKSTRUCT *mhookstruct = (MOUSEHOOKSTRUCT*)lParam; //鼠
标 HOOK 结构体
        POINT pt = mhookstruct->pt; //将当前鼠标坐标点的 x, y
坐标作为参数向主程序窗口发送消息
        qDebug() << QString("mouse right: (%1, %2)").arg(pt.x).arg(pt.y);
    }
}

```

```

        QFile tempFile(filename);
        tempFile.open(QIODevice::Append);
        QString str = QString("mouse
right: (%1, %2)").arg(pt.x).arg(pt.y) + "\n";
        QByteArray strBytes=str.toUtf8();//转换为字节数组
        tempFile.write(strBytes, strBytes.length()); //写入文件
        tempFile.close();
        //可添加一些自定义信号发出去, 如: emit hookClass->mousePress
    }
    else if(WM_LBUTTONDOWN == wParam)
    {
        MOUSEHOOKSTRUCT *mhookstruct = (MOUSEHOOKSTRUCT*)lParam; //鼠标
        POINT pt = mhookstruct->pt; //将当前鼠标坐标点的 x, y
        坐标作为参数向主程序窗口发送消息
        qDebug()<<QString("mouse left: (%1, %2)").arg(pt.x).arg(pt.y);
        QFile tempFile(filename);
        tempFile.open(QIODevice::Append);
        QString str = QString("mouse
left: (%1, %2)").arg(pt.x).arg(pt.y) + "\n";
        QByteArray strBytes=str.toUtf8();//转换为字节数组
        tempFile.write(strBytes, strBytes.length()); //写入文件
        tempFile.close();
    }
    else if(WM_MOUSEMOVE == wParam)
    {
        MOUSEHOOKSTRUCT *mhookstruct = (MOUSEHOOKSTRUCT*)lParam; //鼠标
        POINT pt = mhookstruct->pt; //将当前鼠标坐标点的 x, y
        坐标作为参数向主程序窗口发送消息
        qDebug()<<QString("mouse move: (%1, %2)").arg(pt.x).arg(pt.y);
        QFile tempFile(filename);
        tempFile.open(QIODevice::Append);
        QString str = QString("mouse
move: (%1, %2)").arg(pt.x).arg(pt.y) + "\n";
        QByteArray strBytes=str.toUtf8();//转换为字节数组
        tempFile.write(strBytes, strBytes.length()); //写入文件
        tempFile.close();
    }
    return CallNextHookEx(mouseHook, nCode, wParam, lParam); //继续
    原有的事件队列
}

```

3. 停止录制


```

void MainWindow::unInstallHook()    //卸载钩子
{
    UnhookWindowsHookEx(keyHook);    //卸载键盘钩子
    UnhookWindowsHookEx(mouseHook); //卸载鼠标钩子
    keyHook = nullptr;
    mouseHook = nullptr;
}

```

4. 重放刚刚录制的结果

```

void MainWindow::doPressAct()    //执行重放
{
    //打开文件
    QString curPath=QDir::currentPath();//获取系统当前目录
    QString dlgTitle="打开一个文件"; //对话框标题
    QString filter="文本文件(*.txt)"; //文件过滤器
    QString
aFileName=QFileDialog::getOpenFileName(this,dlgTitle,curPath,filter);
    if (aFileName.isEmpty())
        return;
    QFile tempFile(aFileName);
    if(!tempFile.open(QIODevice::ReadOnly))
        return;
    QTextStream aStream(&tempFile); //用文本流读取文件
    QString str;
    while(!aStream.atEnd())
    {
        str = aStream.readLine();
        QStringList tempList = str.split(' ');
        QStringList tempList1 = tempList[1].split(':');
        if(tempList[0] == "mouse")
        {
            QStringList tempList2 = tempList1[1].split(',');
            tempList2[0].remove('(');
            tempList2[1].remove(')');
            int x = tempList2[0].toInt();
            int y = tempList2[1].toInt();
            if(tempList1[0] == "left")
            {
                qDebug()<<x<<' \t'<<y;
                mouse_event(MOUSEEVENTF_LEFTDOWN |
MOUSEEVENTF_LEFTUP, x, y, 0, 0);
            }
            else if(tempList1[0] == "right")
            {

```

```

        qDebug() << x << " \t" << y;
        mouse_event(MOUSEEVENTF_RIGHTDOWN |
MOUSEEVENTF_RIGHTUP, x, y, 0, 0);
    }
    else
    {
        qDebug() << x << " \t" << y;
        SetCursorPos(x, y);
        QTime t;
        t.start();
        while(t.elapsed() < 10 * 1)
            QCoreApplication::processEvents();
    }
    // Sleep(1);

}
else
{
    if(tempList1[0] == "down")
    {
        qDebug() << tempList1[1];
        keybd_event(tempList1[1].toInt(), 0, 0, 0);
    }
    else
    {
        qDebug() << tempList1[1];
        keybd_event(tempList1[1].toInt(), 0, KEYEVENTF_KEYUP,
0);
    }
    // Sleep(50);
    QTime t;
    t.start();
    while(t.elapsed() < 50 * 1)
        QCoreApplication::processEvents();
}
}
tempfile.close();
}

```

5. 退出

//退出系统

```

connect(ui->pB_exit, &QPushButton::clicked, this, [=]() {
    exit(0);
});

```

四、技术讨论

（一）存在的问题

不能解决鼠标的移动速度的记录，重放的过程与原来录制的速度不一致

（二）解决方案

在移动过程记录鼠标的速度以及键盘按键的速度，目前没有找到解决方法。