基于 Qt 的桌面日历程序

李泽龙

计 54

2015011321

一、 基本功能要求的实现

1. 正确实现按月显示的公历日历：对于此项功能，笔者采用直接复用所给的 demo，即 myCalender 类的方式，实现了此项功能。
2. 可以编辑日历的某一天：点击某一天可以设置这一天的事件，颜色等属性：demo 的代码中提供了将点击 Calender 的某一具体日期与 addNote 的槽的 connect。笔者构建了一个继承于 QDialog 类的 event 对话框，提供了设置这一天的事件、颜色的按钮。
3. 可以添加、编辑、删除某一天的事件序列：在 event 对话框中，最左边为删除单个事件按钮，最右侧提供添加事件、删除所有事件的按钮，即可删除时间序列中的单独一个事件或直接删除整个序列的所有事件。。而后，我将 day_event,week_event,month_event 继承自 Event 基类，并分别用于需要设置每周事件、每月事件时，弹出的窗口。
4. 实现本地文件与桌面日历程序的拖拽交互：控制拖拽交互的开关，通过学习 demo，知道可以通过 setAcceptDrag(bool) 来改变是否能拖拽交互。而后 QCalenderWidget 有三个函数，dragEnterEvent（当用户拖动文件到窗口部件上时候，就会触发 dragEnterEvent 事件）、dropEvent（当用户放下这个文件后，就会触发 dropEvent 事件）、mousePressEvent（当接收到鼠标按下的信号后，就会触发 mousePressEvent 事件），

我将其在派生类实现具体的内容，就可以实现把本地文件拖入桌面日历程序的某一天，在当天的格子内显示文件名称，并且把文件存入桌面日历程序的某个文件夹下，并通过日历程序把存放在某天的文件通过拖拽文件名称存放到本地文件夹内进行保存。。

5. 支持使用配置文件（如 XML 文件）进行数据同步：笔者因为不会使用 XML 文件。因而通过生成文本文件.event，并添加导入导出按钮，通过信号和槽实现使用配置文件进行数据同步。
6. 可以对日历进行整体拖拽和固定&鼠标穿透：固定和拖拽通过设置 MainWindow 实例的状态栏的隐藏和出现来控制日历是否能整体拖拽和固定；鼠标穿透：通过调整 MainWindow 的结构层次，来设置鼠标是否能穿透日历。
7. 国际化:给所有需要翻译的文本加上 tr(),pro 文件中加入：TRANSLATIONS+=tr_chinese.ts 然后在命令行执行以下操作：

```
C:\Qt\Project-Week1_722404375\Project-Week1\lz1_calendar>lupdate lz1_calendar.pro
Updating 'tr_chinese.ts'...
    Found 51 source text(s) (51 new and 0 already existing)

C:\Qt\Project-Week1_722404375\Project-Week1\lz1_calendar>linguist

C:\Qt\Project-Week1_722404375\Project-Week1\lz1_calendar>lrelease tr_chinese.ts
Updating 'tr_chinese.qm'...
    Generated 51 translation(s) (51 finished and 0 unfinished)

C:\Qt\Project-Week1_722404375\Project-Week1\lz1_calendar>
```

其中，lupdate *.pro 生成.ts 文件；通过 linguist 翻译文本；lrelease *.ts 生成.qm 文件，最后将.qm 文件配置进

QT Creator IDE 的资源中，并建立一个 QTranslator 的实例，即可完成中英文的选择。当然，笔者还加了一个日本语的选项，有兴趣可以点一点试试☺

二、 系统结构

```
class event_data //事件存储
{
public:
    event_data();
    ~event_data();
    int cnt; //事件个数
    QVector <QString> memo_data; //存储事件
    QVector <QTime> moment_data; //存储时间
    void add(QTime time, QString memo); //添加事件
    void output(QTextStream& s); //导入事件到文件
};

class day_data : public event_data //日期时间存储
{
public:
    day_data(QDate date = QDate(), QColor color = Qt::white);
    ~day_data();
```

```
        QDate cur; //日期
        QColor grid_color; //日期格子颜色
        void output(QTextStream& s); //导入事件到文件

    };

    class month_data : public event_data //月份存储
    {
    public:
        month_data(QDate date = QDate());
        ~month_data();
        int year, month; //事件的年月
        void output(QTextStream& s);

};
    class Event : public QDialog //编辑对话框
    {
        Q_OBJECT
    public:
        Event(QWidget *parent = NULL);
        ~Event();
        QPushButton    *add_event,    *delete_event[5],
```

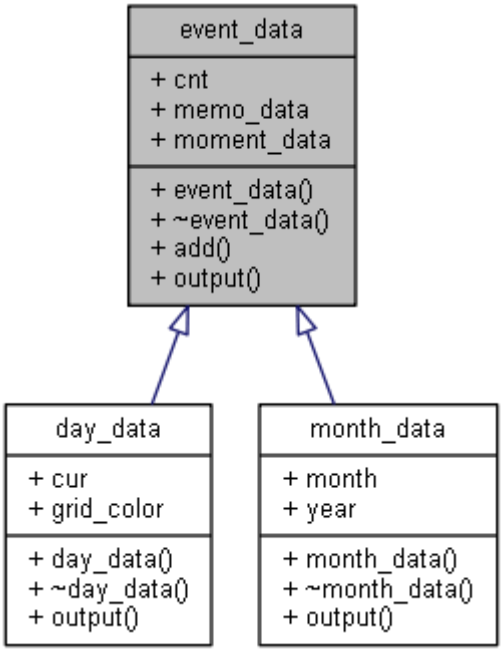
```
*delete_all, *set_color; //按钮：添加事件、删除事件、删除所有事件、设置颜色
        QLabel* moment[5], *note[5]; //显示时间和事件

    public slots:
        virtual void maintain() = 0; //维护事件数据并显示
        virtual void check() = 0; //检查事件数量并添加事件
        virtual void deleteAll() = 0; //删除所有事件
        virtual void deleteEvent(const int index) = 0; //删除单个事件
    };

    /*********************************************************************************/

    class day_event : public Event
    {
        Q_OBJECT
    public:
        day_event(QDate date, QCalendarWidget *parent
```

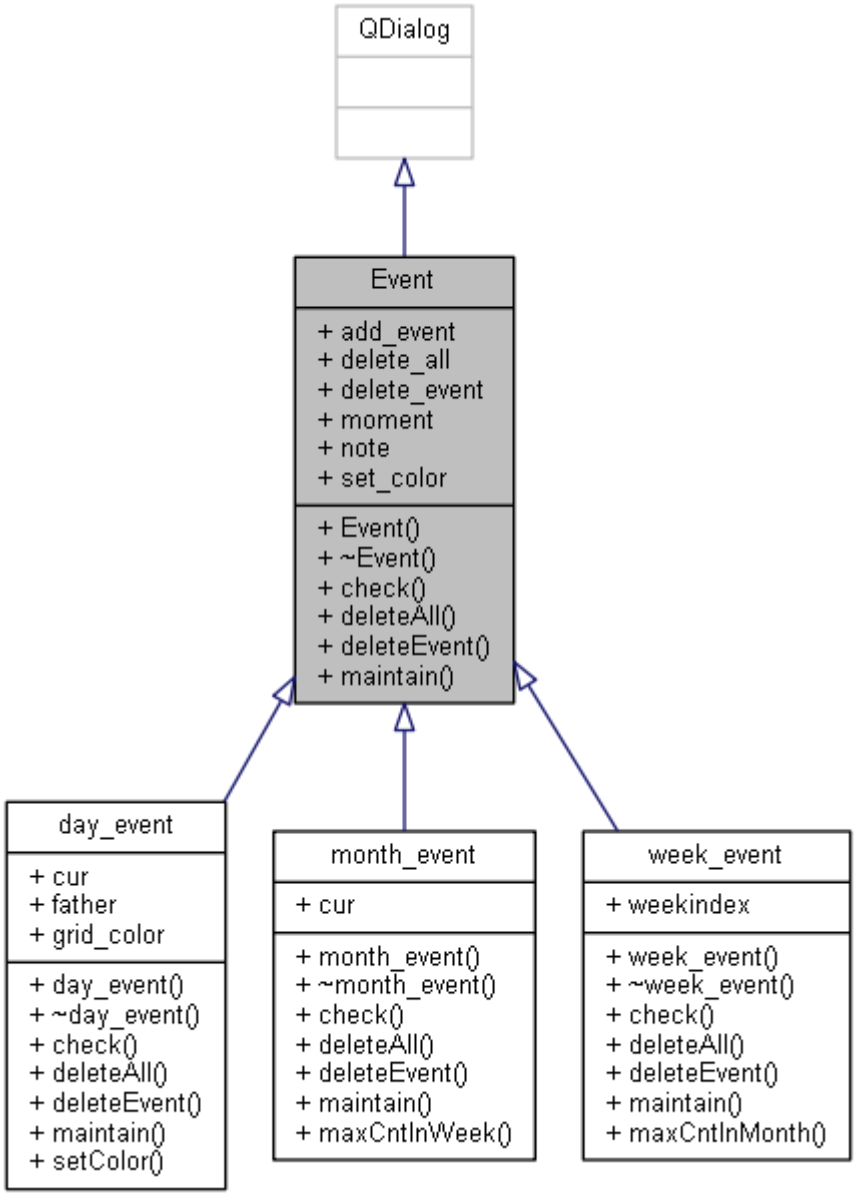
```
= NULL);
        ~day_event();
        QDate cur;
        QBrush grid_color;
        QCalendarWidget *father; //编辑日期格子颜色用

    public slots:
        void check();
        void deleteAll();
        void deleteEvent(const int index);
        void setColor();
        void maintain();
    };

    /************************************************
*****************************************/

    class week_event : public Event
    {
        Q_OBJECT
    public:
        week_event(int index, QWidget *parent = NULL);
```

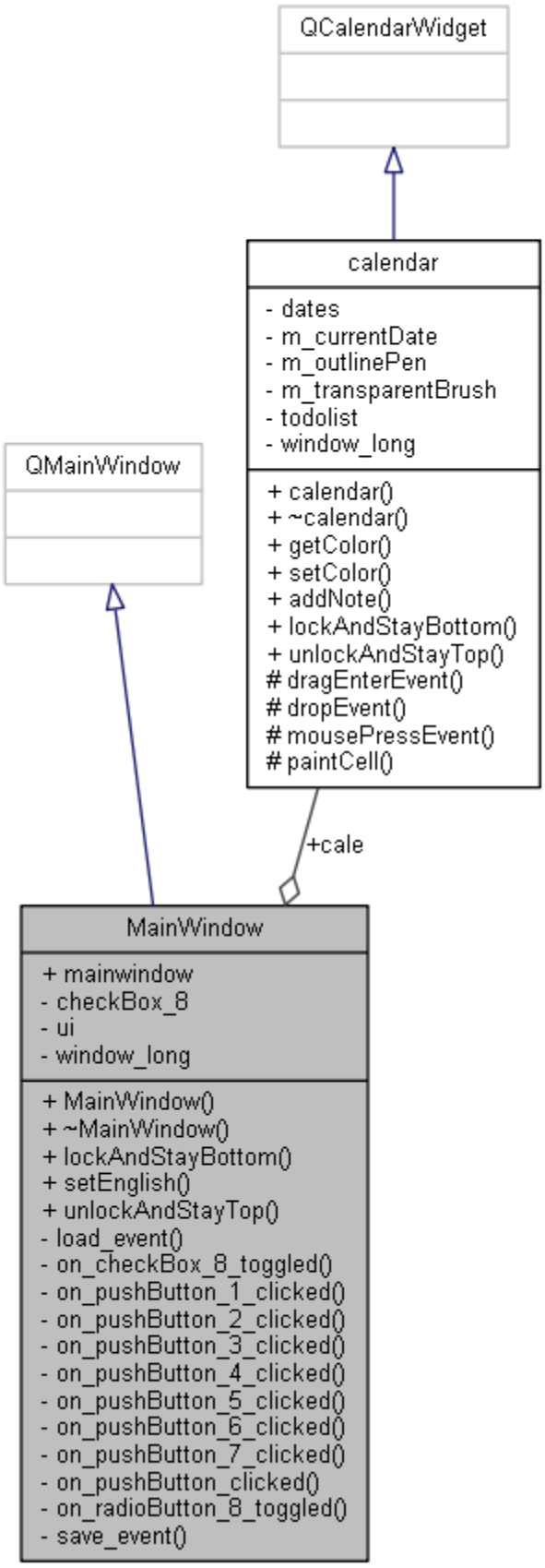
```
    ~week_event() {}
    int weekindex;

public slots:
    void maintain();
    void check();
    void deleteAll();
    void deleteEvent(const int index);
    int maxCntInMonth(); //一个月最大的事件数
};

/*************************************************
*****************************************/

class month_event : public Event
{
    Q_OBJECT
public:
    month_event(QDate date, QWidget *parent = NULL);
    ~month_event() {}
    QDate cur;
```

```
    public slots:
        void maintain();
        void check();
        void deleteAll();
        void deleteEvent(const int index);
        int maxCntInWeek(); //一周最大的事件数
};
```

三、 一些结构图




QDialog
Event
+ add_event
+ delete_all
+ delete_event
+ moment
+ note
+ set_color
+ Event()
+ ~Event()
+ check()
+ deleteAll()
+ deleteEvent()
+ maintain()
day_event
+ cur
+ father
+ grid_color
+ day_event()
+ ~day_event()
+ check()
+ deleteAll()
+ deleteEvent()
+ maintain()
+ setColor()
month_event
+ cur
+ month_event()
+ ~month_event()
+ check()
+ deleteAll()
+ deleteEvent()
+ maintain()
+ maxCntInWeek()
week_event
+ weekindex
+ week_event()
+ ~week_event()
+ check()
+ deleteAll()
+ deleteEvent()
+ maintain()
+ maxCntInMonth()


QCalendarWidget
calendar
- dates
- m_currentDate
- m_outlinePen
- m_transparentBrush
- todolist
- window_long
+ calendar()
+ ~calendar()
+ getColor()
+ setColor()
+ addNote()
+ lockAndStayBottom()
+ unlockAndStayTop()
# dragEnterEvent()
# dropEvent()
# mousePressEvent()
# paintCell()
QMainWindow
+cale
MainWindow
+ mainwindow
- checkBox_8
- ui
- window_long
+ MainWindow()
+ ~MainWindow()
+ lockAndStayBottom()
+ setEnglish()
+ unlockAndStayTop()
- load_event()
- on_checkBox_8_toggled()
- on_pushButton_1_clicked()
- on_pushButton_2_clicked()
- on_pushButton_3_clicked()
- on_pushButton_4_clicked()
- on_pushButton_5_clicked()
- on_pushButton_6_clicked()
- on_pushButton_7_clicked()
- on_pushButton_clicked()
- on_radioButton_8_toggled()
- save_event()