数据结构小组汇报

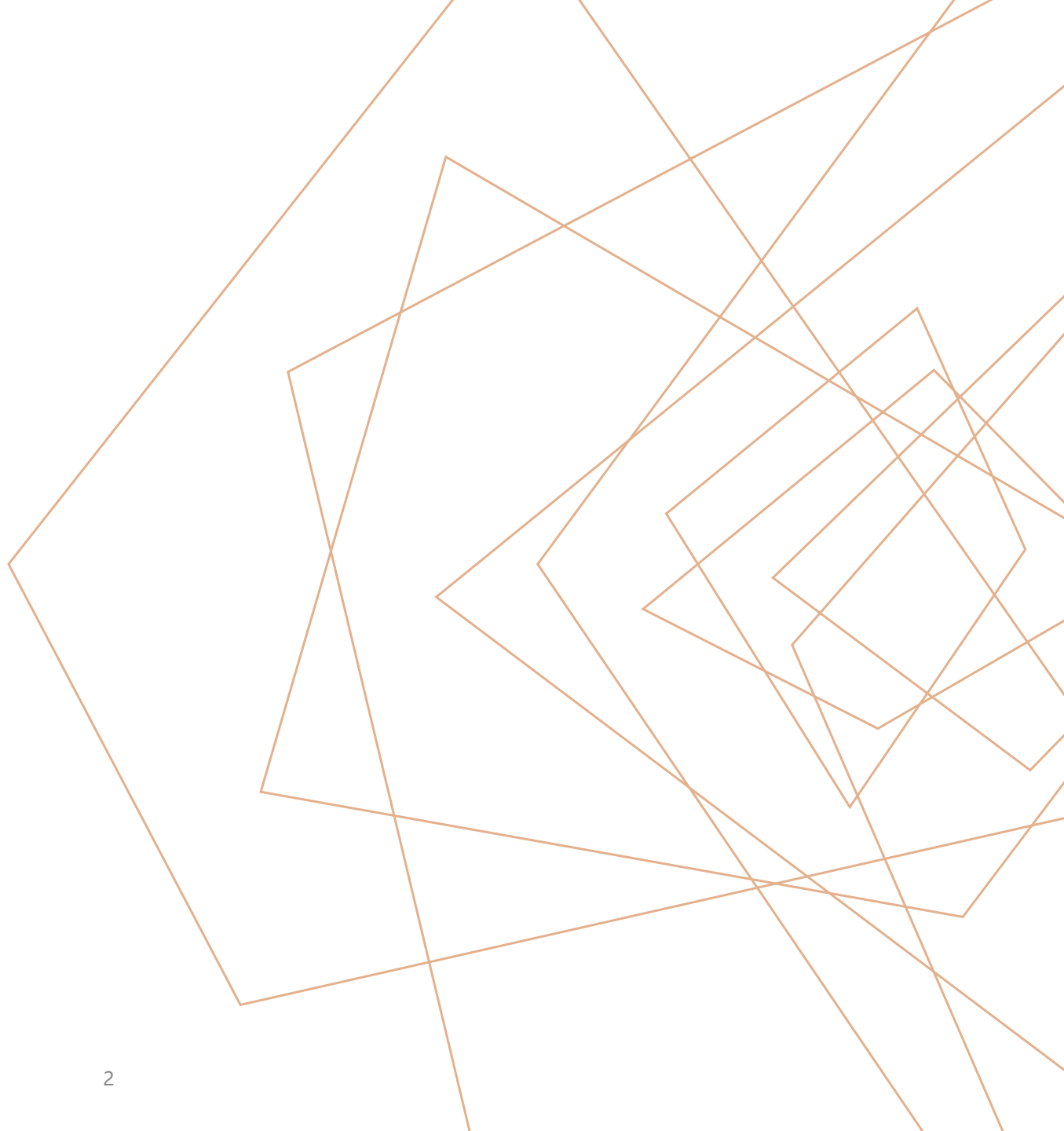
汇报人：薛维浩
组员： 王珍 刘清 HAMIDOU

项目简介

内容： 21点游戏

使用Qt（跨平台的C++框架）

IDE : Qt Creator



项目实现21点

1. 21点基本逻辑

使用一个card对象表示卡牌，其中包含数字和花色

使用线性表结构表征牌组

使用时间函数初始化随机数，对牌组仅限打乱

按顺序发给庄家和玩家

结算时利用函数计算各家分值进行比较

2. UI界面实现

依次使用Qwidget，QGraphicsView，QGraphicsPixmapItem，QPixmap等Qt提供的对象，以窗口，界面，场景，物品，图片的顺序，创建对象树。

分模块实现图片加载，物品放置，场景切换，界面生成等函数，

然后在游戏的3个阶段中分别调用

项目实现21点

3. 交互设计

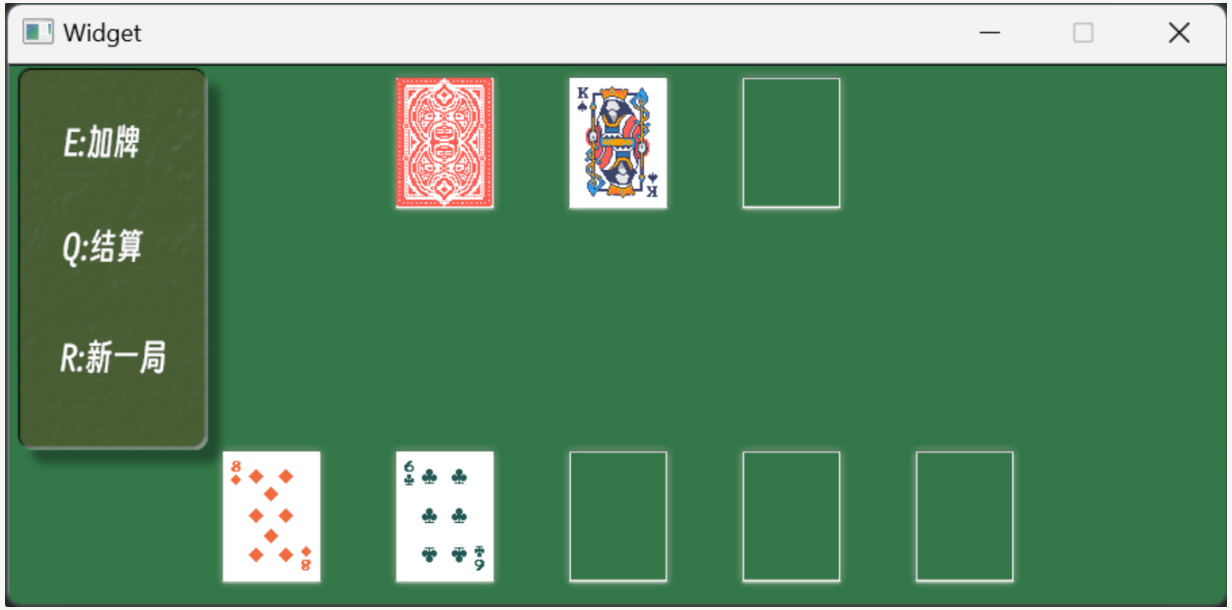
信号与槽，通过将各类事件创建的信号函数，对槽函数进行触发，实现对鼠标和键盘的部分监视

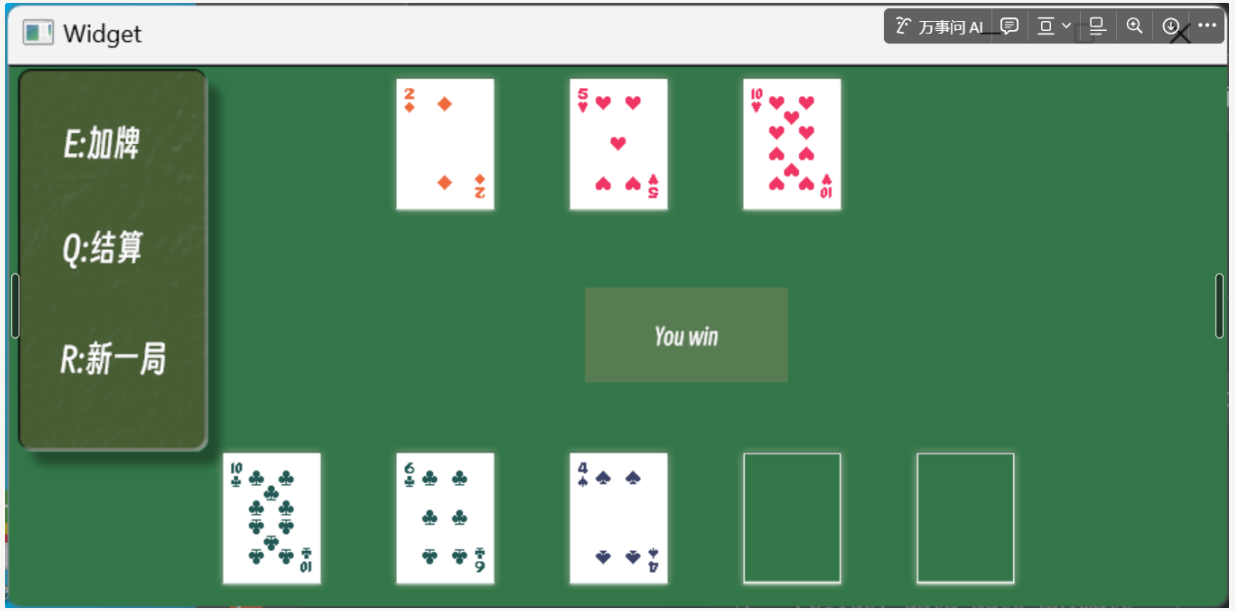
```
void Widget::keyPressEvent(QKeyEvent* event)
{
    keyPressed = event->key();

    if( keyPressed == Qt::Key_E)
    {
        emit keyBoardpressE(keyPressed);
    }
    if( keyPressed == Qt::Key_Q)
    {
        emit keyBoardpressQ(keyPressed);
    }
    if( keyPressed == Qt::Key_R){
        emit keyBoardpressR(keyPressed);
    }
}
```

```
void Widget::initmemuscene(){
    memu.setSceneRect(QRect(0,0,900,400));
    QGraphicsPixmapItem *mBackground = new QGraphicsPixmapItem;
    mBackground->setPixmap(QPixmap(":/image/D:/desktop/img/startBackground.png"));
    memu.addItem(mBackground);
    QPushButton *button = new QPushButton("开始游戏");
    QGraphicsProxyWidget *proxy = memu.addWidget(button);
    proxy->setPos(420,180);
    mGameView.setScene(&memu);
    connect(button,&QPushButton::clicked,this,&Widget::initgamescene);

}
```





受限于时间因素，UI制作较为简陋

要点

模块化编程 —— 将程序分解为独立的、可重用的模块，每个模块负责特定的功能。

内存管理 —— 自动释放子对象内存，防止内存泄漏。

对象通信 —— 独特的信号与槽机制使得对象之间的通信更加灵活和高效

其他 —— 使用各类高效算法

模块化编程

将各个功能，用不同的函数实现，例如：界面初始化函数，图片载入函数，音乐播放函数等等

利于程序的进一步扩展，以及各种功能的微调

```
{
    QGraphicsPixmapItem* cardK = new QGraphicsPixmapItem;
    cardK->setPixmap(load_card(num,s));
    cardK->setPos(x,y);
    mScene.addItem(cardK)
}

void Widget::placeStatue(
    QGraphicsPixmapItem*                                    xmapItem;
    QPixmap* image = new
    if(num ==1){
        // image->load(":/image/D:/desktop/img/testimage.png");
```

function load_card

→ QPixmap

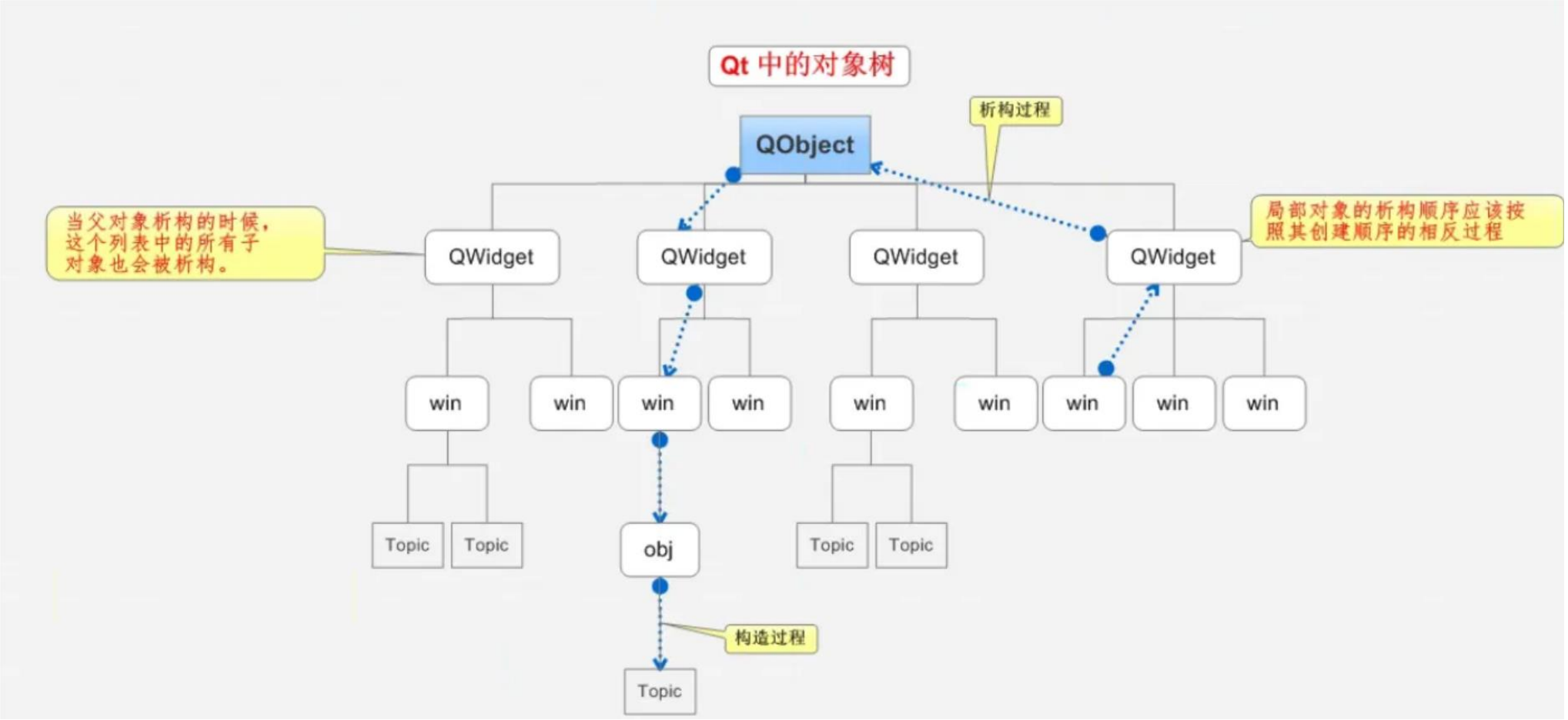
Parameters:

- int num
- color s

返回一个QPixmap的指针 num:牌号 s 牌型

```
QPixmap load_card(int num, color s)
```

内存管理-对象树


Qt 中的对象树
析构过程
QObject
当父对象析构的时候，
这个列表中的所有子
对象也会被析构。
局部对象的析构顺序应该按
照其创建顺序的相反过程
QWidget
QWidget
QWidget
QWidget
win
win
win
win
win
win
win
win
win
Topic
Topic
obj
Topic
Topic
构造过程
Topic

代码

QT的对象可以设置自己的父对象，形成一个对象树

通过对象树方便的管理和释放内存，每一个对象都初始化出其父类对象，在父类对象析构后，自动析构子对象

```
class Widget : public QWidget
{
    Q_OBJECT
    ...
}
.cpp
Widget::Widget(QWidget *parent)
{
    ...
}
```

Widget对象的定义实例

对象通信

```
signals:
    // 定义一个信号，当 'E' 键按下时发出
    void keyBoardpressE(int key);

    void keyBoardpressQ(int key);

    void keyBoardpressR(int key);

    void playmusic(int time,int num);

    void cardLimit(int num);

    void theEnd();
```

信号与槽

信号与槽 是 QT 框架中的一种通信机制，用于对象之间的交互。信号（SIGNAL）是事件触发的通知，槽（SLOT）是响应信号的函数。通过 CONNECT 函数将信号与槽绑定，当信号发出时，槽函数会自动执行。信号与槽是类型安全的，支持多对多连接，且可以跨线程使用。它是 QT 的核心特性，取代了传统的回调函数，使代码更清晰、灵活和易于维护。

```
connect(this,&Widget::keyBoardpressE,this,&Widget::gameContinue);
connect(this,&Widget::keyBoardpressQ,this,&Widget::gameOver);
connect(this,&Widget::playmusic,this,&Widget::playMedie);
connect(this,&Widget::cardLimit,this,&Widget::checkCard);
connect(this,&Widget::keyBoardpressR,this,&Widget::gameRestart);
```

谢谢