# 27 坐标系统和坐标变换

## 27.1 坐标变换方法

QPainter在窗口绘图区中绘制图形的默认坐标系，原点位于绘图区的左上角，X轴向右为正，Y轴向下为正，1代表1像素。这样的坐标系也叫设备坐标系。此外，QPainter还提供了一些坐标变换的功能，将设备坐标系平移、缩放、旋转、扭曲为逻辑坐标系。很多时候，在逻辑坐标系中绘制图形，比使用设备坐标更加方便。

QPainter类提供了一套与坐标变换有关的方法：

```
void QPainter::translate(qreal dx, qreal dy); // 坐标系平移
void QPainter::rotate(qreal angle);           // 坐标系旋转
void QPainter::scale(qreal sx, qreal sy);     // 坐标系缩放
void QPainter::shear(qreal sh, qreal sv);     // 坐标系扭曲
void QPainter::save();                        // 将当前坐标系压入到堆栈中
void QPainter::restore();                     // 从堆栈中弹出并恢复坐标系
void QPainter::resetTransform();              // 恢复到默认坐标系
```

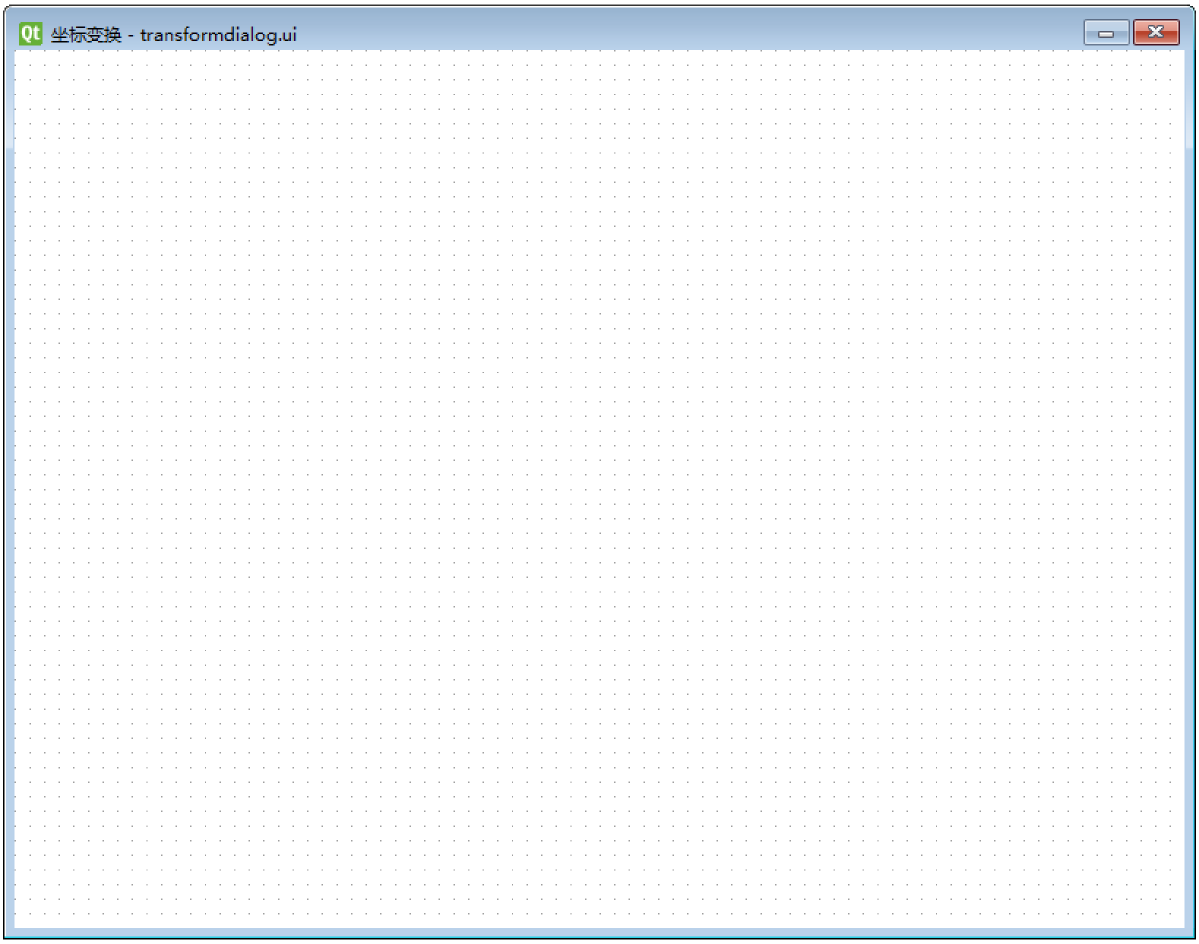
此外，QPainter类也支持通过坐标变换矩阵（CTM）变换坐标系。

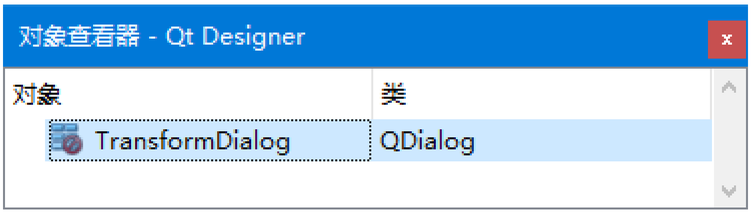
## 27.2 案例

### 27.2.1 创建项目

通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Transform的项目。

### 27.2.2 设计界面





对象查看器 - Qt Designer

| 对象 | 类 |
|---|---|
| TransformDialog | QDialog |

C:\Users\Minwei\Projects\Qt\Transform\transformdialog.ui：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>TransformDialog</class>
 <widget class="QDialog" name="TransformDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Window">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </active>
    <inactive>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Window">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
```

```
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </inactive>
    <disabled>
     <colorrole role="Base">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Window">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </disabled>
   </palette>
  </property>
  <property name="windowTitle">
   <string>坐标变换</string>
  </property>
 </widget>
 <resources/>
 <connections/>
</ui>
```

### 27.2.3 实现功能

C:\Users\Minwei\Projects\Qt\Transform\transformdialog.h：

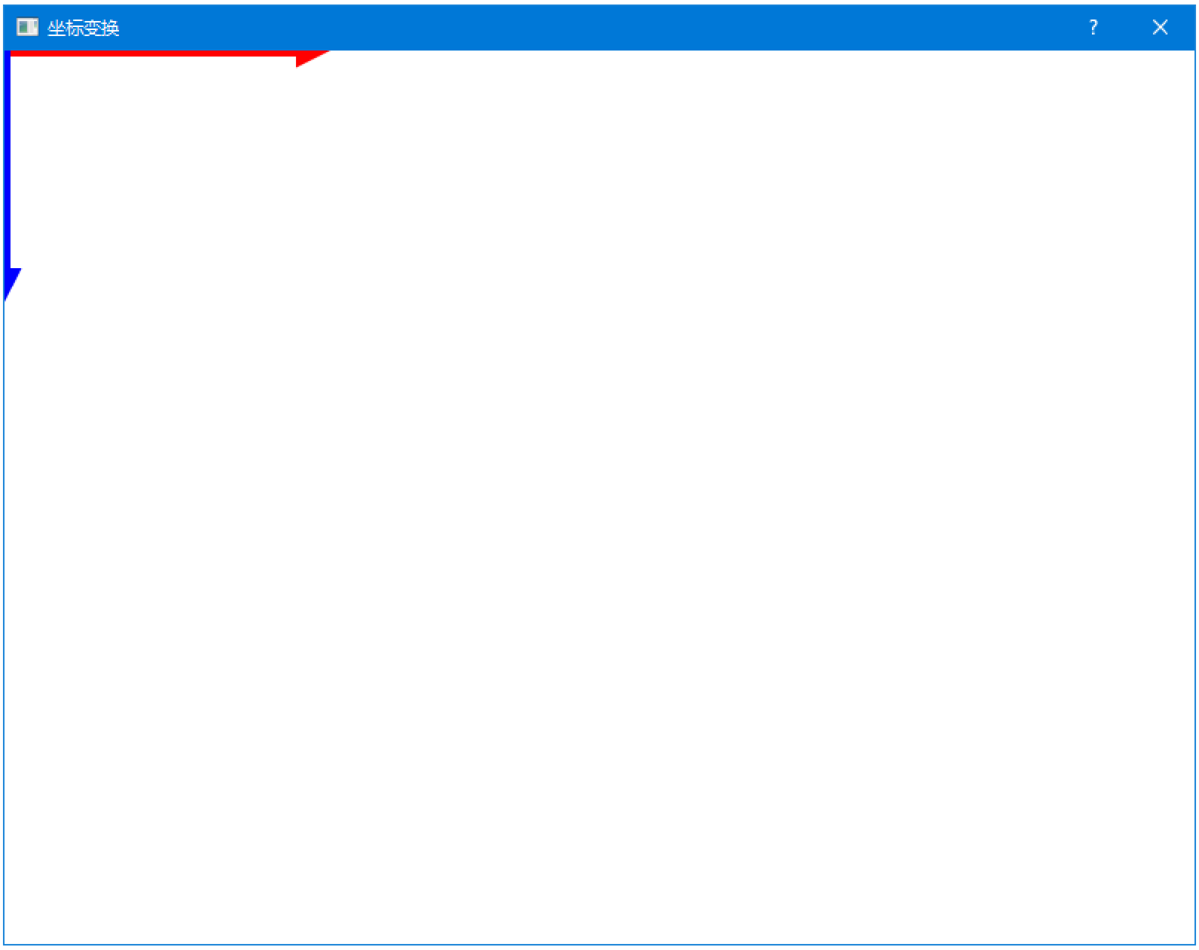
```
#ifndef TRANSFORMDIALOG_H
#define TRANSFORMDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class TransformDialog; }
QT_END_NAMESPACE

class TransformDialog : public QDialog
{
    Q_OBJECT

public:
    TransformDialog(QWidget *parent = nullptr);
    ~TransformDialog();

protected:
    void paintEvent(QPaintEvent*);

private:
    Ui::TransformDialog *ui;
};

#endif // TRANSFORMDIALOG_H
```

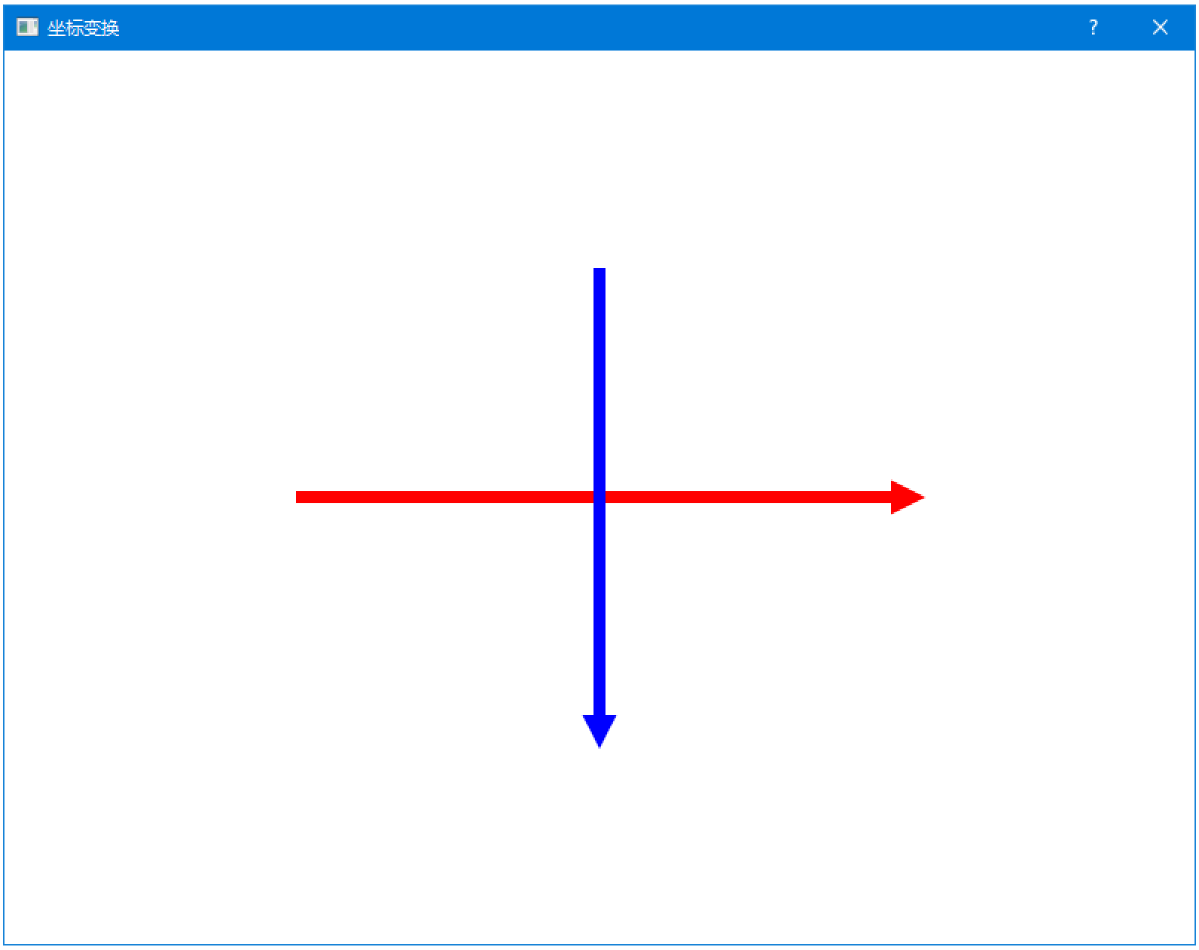
C:\Users\Minwei\Projects\Qt\Transform\transformdialog.cpp:

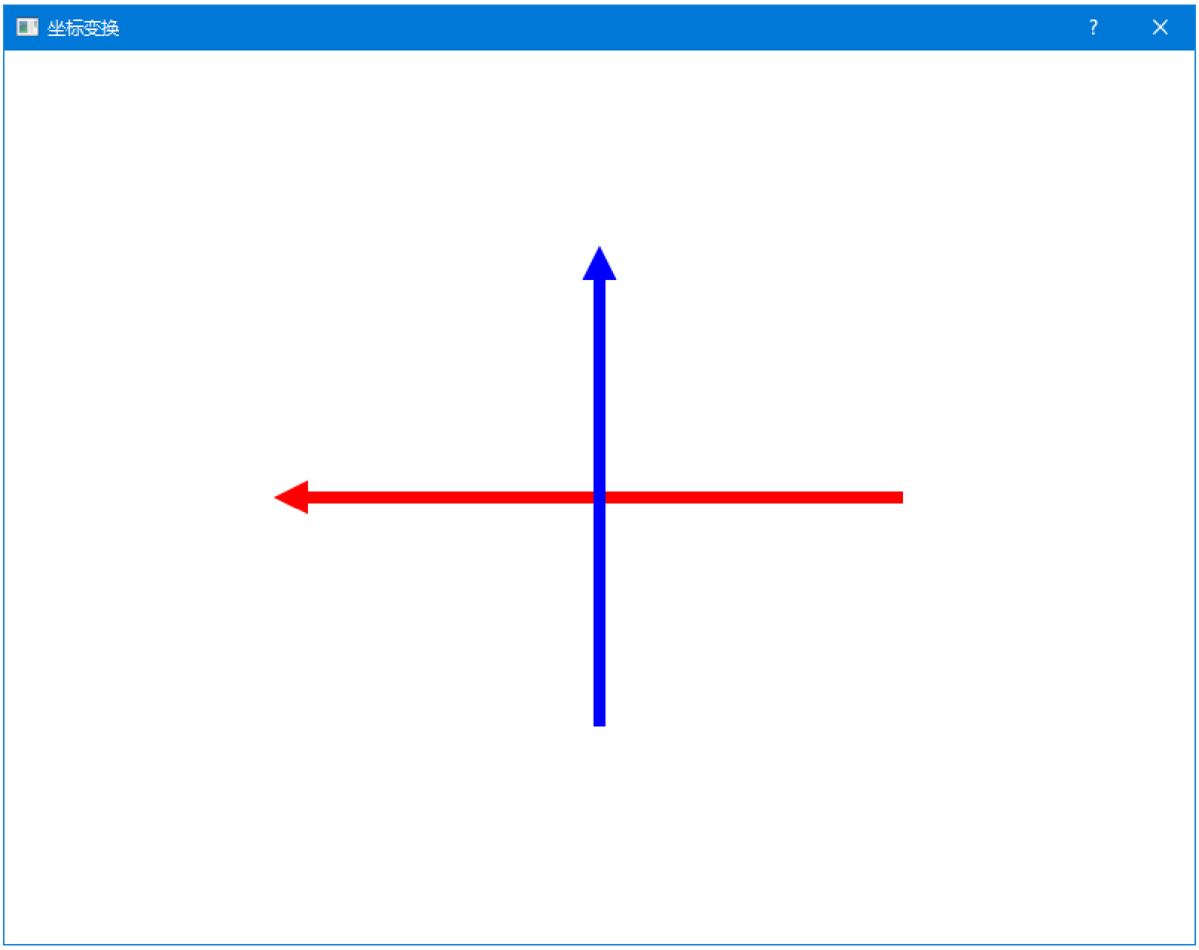
```
#include <QPainter>

#include "transformdialog.h"
#include "ui_transformdialog.h"

TransformDialog::TransformDialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::TransformDialog)
{
    ui->setupUi(this);
}

TransformDialog::~TransformDialog()
{
    delete ui;
}

void TransformDialog::paintEvent(QPaintEvent*)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    int w = width(), h = height();

    painter.translate(w/2, h/2);
    painter.rotate(180);
    painter.scale(-1.5, 1.5);
    painter.shear(1, 0);
    //painter.resetTransform();

    QPen pen;
    pen.setWidth(8);
    pen.setStyle(Qt::SolidLine);
    pen.setJoinStyle(Qt::MiterJoin);

    pen.setColor(Qt::red);
    painter.setPen(pen);
    painter.drawLine(QPoint(-w/4, 0), QPoint(w/4, 0));
    painter.drawPolygon(QPolygon()
        << QPoint(w/4, -5) << QPoint(w/4+10, 0) << QPoint(w/4, 5));

    pen.setColor(Qt::blue);
    painter.setPen(pen);
    painter.drawLine(QPoint(0, -h/4), QPoint(0, h/4));
    painter.drawPolygon(QPolygon()
        << QPoint(-5, h/4) << QPoint(0, h/4+10) << QPoint(5, h/4));
}
```

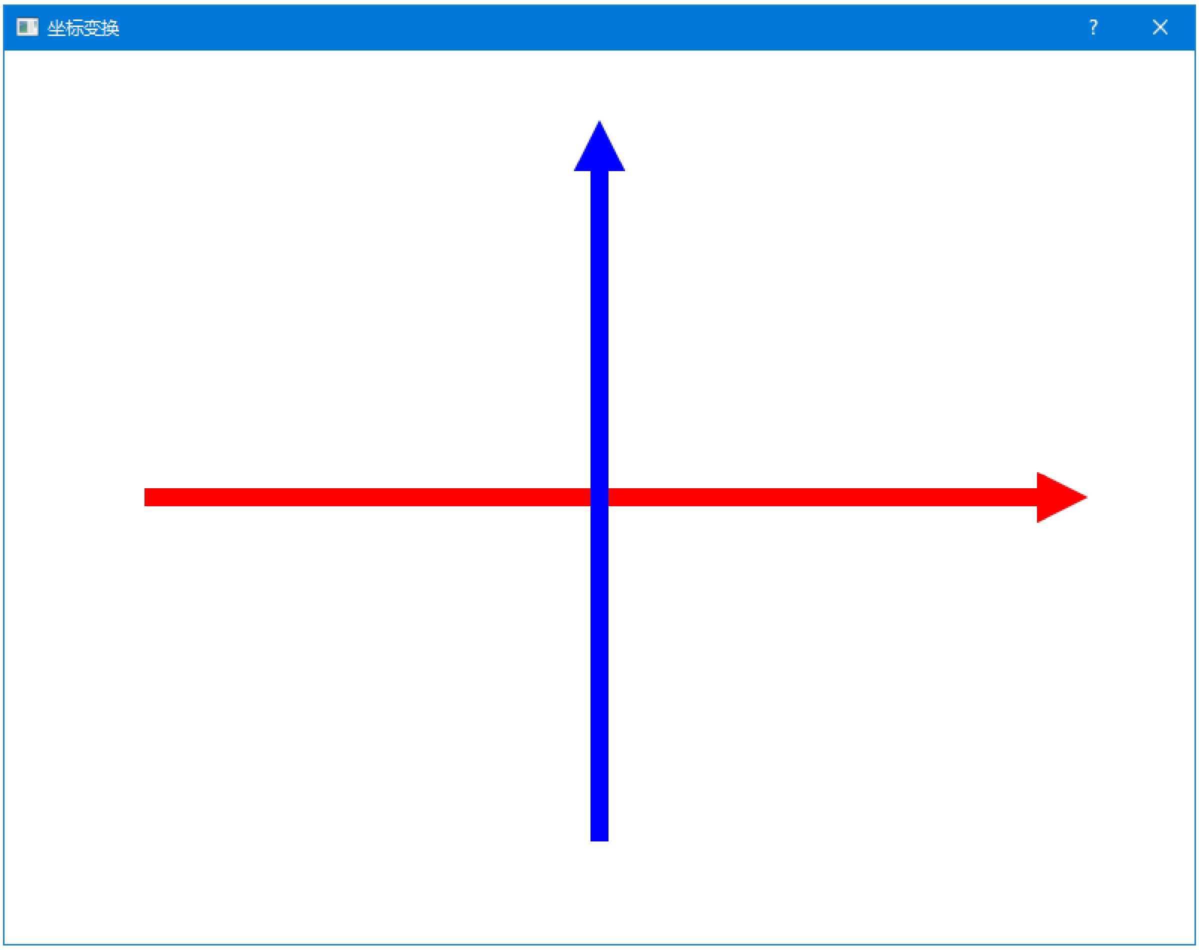
### 27.2.4 测试验证

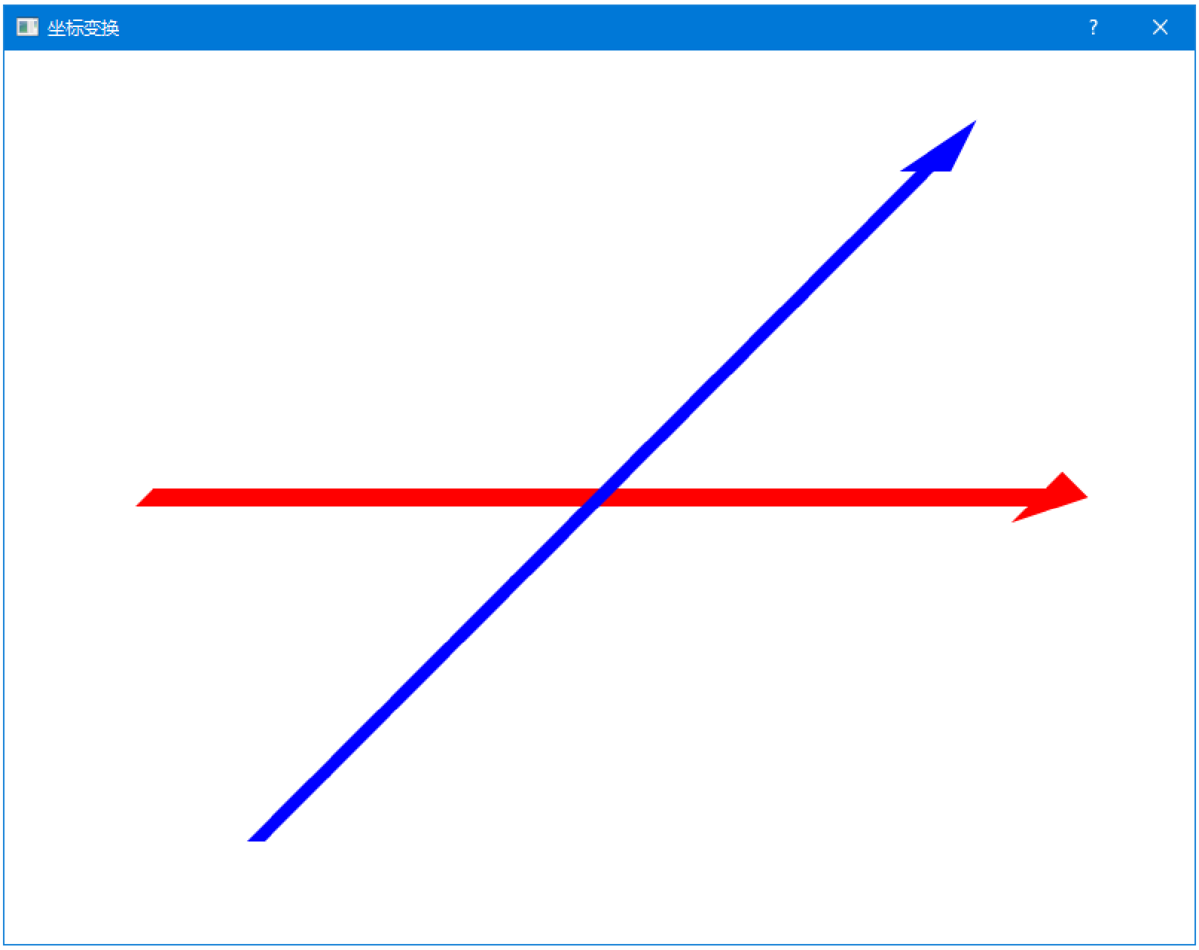
运行效果如图所示：

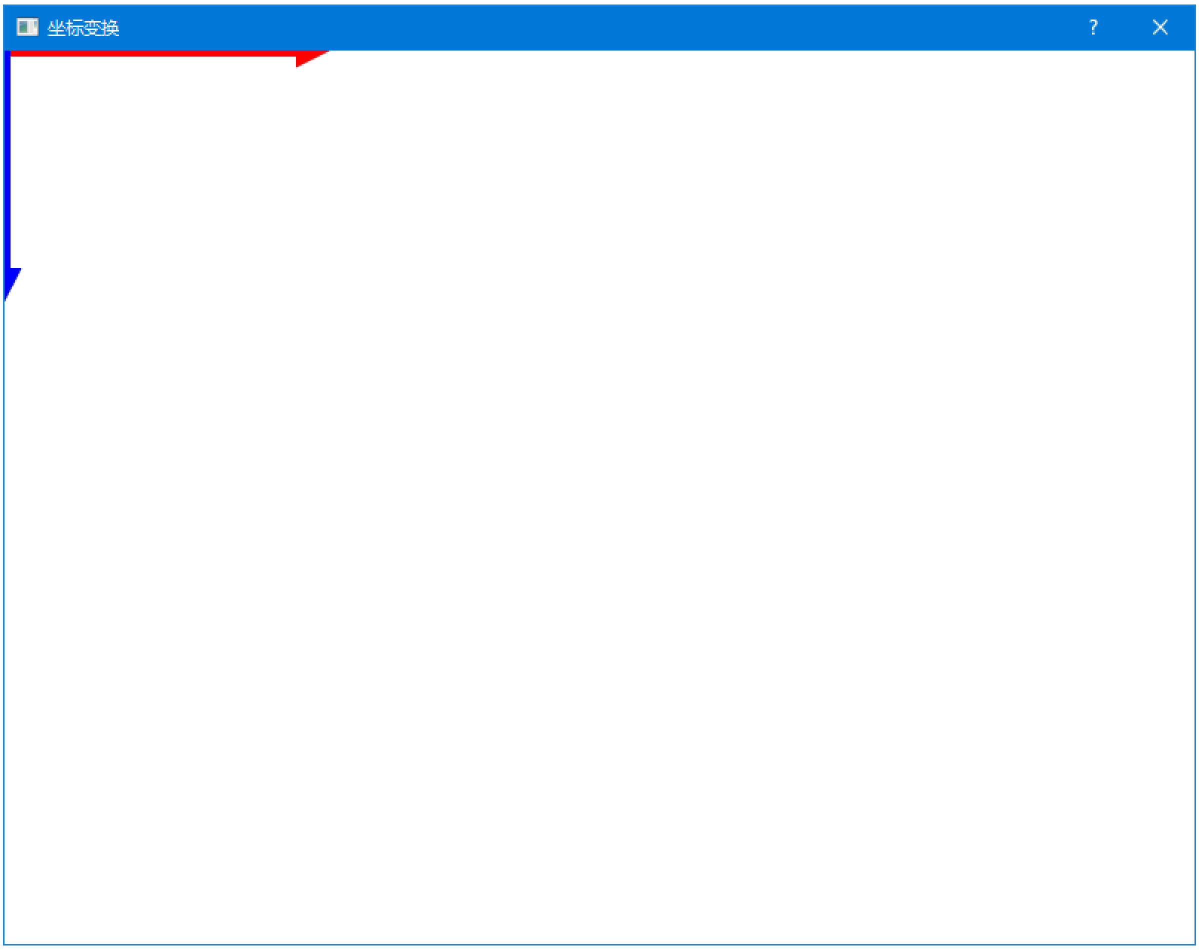



坐标变换
?


坐标变换
?


坐标变换
?
×


坐标变换
?
×



坐标变换
?