QPainte的使用必须在paintEvent()函数中

- 在其他函数中使用会报错（画图事件函数会自动调用，自己手动调用会报错）

```
QWidget::paintEngine: Should no longer be called//已经被调用过了，不需要再调用
QPainter::begin: Paint device returned engine == 0, type: 1
```

绘图

- 需要三个对象**QPainter**、**QPen**、**QBrush**
- 创建绘图设备->创建画笔设置画笔->创建笔刷设置笔刷->画线
- 常用**drawLint/drawArc/drawRect/drawEllipse/drawPolygon**

```
void Widget::paintEvent(QPaintEvent*e){
 //使用了构造函数，并指定在Widget设备上绘图
 QPainter painter(this);
 //如果使用不带参数的构造函数，则需要自定义绘图设备
 QPainter painter;
 painter.begin(this);
 painter.end();

 //创建画笔(画刷，线宽，画笔风格，画笔端点风格，画笔连接风格)
  QPen pen(Qt::green,5,Qt::DotLine,Qt::RoundCap,Qt::RoundJoin);
  pen.setWidth(1);//重新定义宽度
  pen.setStyle(Qt::SolidLine);//画笔风格实线
  //使用画笔
  painter.setPen(pen);

  //QBrush类提供画刷填充图形（颜色，风格），
  //颜色可以使用QColor类或枚举类Qt::BrushStyle
  QBrush brush(QColor(0,0,255),Qt::Dense4Pattern);
  painter.setBrush(brush);

  //两点绘制直线
  painter.drawLine(QPoint(0,0),QPoint(100,100));
```

```
23
24  //绘制矩形，浮点型（坐标x，坐标y，矩形宽，矩形高）
25  QRectF rectangle (70.0,40.0,80.0,60.0);
26  QRect rectangle (70,40,80,60);
27  painter.drawRect(QRectF);
28
29  //绘制圆弧
30  int starAngle=30*16;//起始角 =实际角度*16，0°指向时钟表3点得位置
31  int spanAngle=120*16;//逆时针为正，顺时针为负
32  painter.drawArc(rectangle,starAngle,spanAngle);
33  painter.drawEllipse(220,20,50,50);
34
35  //定义4点 浮点型
36  static const QPointF points[4]={
37  QPointF(270.0,80.0),
38  QPointF(290.0,10.0),
39  QPointF(350.0,30.0),
40  QPointF(390.0,70.0)
41  };
42  painter.drawPolygon(points,4);//需要指定顶点，顶点个数，填充规则
43
44  //使用画刷填充一个矩形区域
45  painter.fillRect(QRect(10,100,150,20),QBrush(Qt::darkYellow));
46
47  //擦除一个矩形区域的内容
48  painter.eraseRect(QRect(50,0,50,120));
49 }
```

坐标系统

- **QPointer**提供了坐标系统的变化
- 在变化坐标之前常使用**save()**和**restore()**保存当前坐标系统和恢复
- 对坐标系统的操作在**save**和**restore**之间进行
- 常用操作：**translate()/rotate()/scale()/shear()**
- 打开抗锯齿：**setRenderHint(QPainter::Antialiasing);**

```
1 void Widget::paintEvent(QPaintEvent *event){
```

```
QPainter painter(this);
//填充背景为白色
painter.fillRect(rect(),Qt::white);
painter.setPen(QPen(Qt::red,11));
//绘制线段
painter.drawLine(QPoint(5,6),QPoint(100,99));
//坐标系统平移
painter.translate(200,150);
 //开启抗锯齿
 painter.setRenderHint(QPainter::Antialiasing);
 //再绘制一条
 painter.drawLine(QPoint(5,6),QPoint(100,99));

 //先保存当前坐标系统
 painter.save();
 //逆时针旋转坐标系统90°
 painter.rotate(90);
 painter.setPen(Qt::cyan);
 painter.drawLine(QPoint(5,6),QPoint(100,99));
 painter.restore();

 QPen pen;
 painter.setPen(pen);//使用默认画笔
 painter.setBrush(Qt::darkGreen);
 //绘制一个矩形
 painter.drawRect(-50,-50,100,50);

 painter.save();
 //将坐标系进行缩放
 painter.scale(0.5,0.4);//（水平方向，垂直方向）
 painter.setBrush(Qt::yellow);
 painter.drawRect(-50,-50,100,50);
 painter.restore();

 painter.save();
```

```
 painter.setPen(Qt::blue);
 painter.setBrush(Qt::darkYellow);
 painter.drawEllipse(QRect(60,-100,50,50));
 //坐标系扭曲
 painter.shear(1.5,-0.7);//（水平方向扭曲值，垂直方向）
 painter.setBrush(Qt::darkGray);
 painter.drawEllipse(QRect(60,-100,50,50));
 painter.restore();
}
```

视口与窗口的转换

```
void Widget::paintEvent(QPaintEvent *event){
 QPainter painter(this);
 //逻辑坐标的（-50，-50）对应物理坐标的（0，0）
 painter.setWindow(-50,-50,100,100);
 painter.setBrush(Qt::green);
 //此时的矩形会发生变形，
 painter.drawRect(0,0,20,20);
}

//为了防止变形，需要将视口（物理坐标）和窗口（逻辑坐标） 的宽和高对应比例设置成相同值
void Widget::paintEvent(QPaintEvent *event)
{
 QPainter painter(this);
 //获得窗口长宽的最小值
 int size=qMin(width(),height());
 int x= width()/2;
 int y= height()/2;
 //设置视口，视口的坐标 size=600
 painter.setViewport(x,y,size,size);
 painter.setWindow(0,0,100,100);//窗口是正方形
 painter.setBrush(Qt::green);
```

```
22   //在窗口上画，但映射到视口上，比例放大了6呗，在视口上变成rect(0,0,1
 20,120)
23    painter.drawRect(0,0,20,20);
24  }
```

实现钟表：定时器和绘图相结合

- **#include<QTimer>**包含定时器头文件，需要一个**QTimer*timer**成员变量，构造函数中定时调用**update()**函数
- **QTransform transform**转换对象，用于转换坐标系统

```
1  void Widget::paintEvent(QPaintEvent*event){
2   angle+=10;
3   if(angle==360)angle=0;
4   int side=qMin(width(),height());
5
6   QPainter painter(this);
7   painter.setRenderHint(QPainter::Antialiasing);//抗锯齿
8
9   //定义一个转换对象，
10   QTransform transform;
11   transform.translate(width()/2,height()/2);//坐标移动至中心
12   transform.scale(side/300,side/300);//缩放成300*300的正方形
13   transform.rotate(angle);
14   painter.setWorldTransform(transform);//变换所以变化
15
16   //(x,y)是矩形的左上角坐标，(w,h)是宽高
17   painter.drawEllipse(-150,-150,300,300);
18   painter.drawLine(0,0,100,0);
19  }
```