

Классы для рисования и работы с графическим интерфейсом AgavaSCADA/AgavaPLC



Содержание

[Обзор](#)

[Основные классы и структуры](#)

[Rect](#)

[Point](#)

[Color](#)

[Font](#)

[Pen](#)

[ResourceLink](#)

[Gradient](#)

[Brush](#)

[Alignment](#)

[Класс Painter](#)

[Методы рисования фигур](#)

[Методы заливки и очистки](#)

[Управление состоянием](#)

[Трансформации и clipping](#)

[Вспомогательные методы](#)

[Полный пример использования Painter](#)

[Обработка событий](#)

[MouseEvent](#)

[WheelEvent](#)

1 Обзор

Данное описание предназначено для использования с системами версии 1.6+. Ниже описываются инструменты, реализующие вывод графических примитивов, инструменты рисования и обработки событий.

2 Основные классы и структуры

2.1 Rect

Прямоугольная область с координатами и размерами.

Конструкторы:

- `Rect()` - пустой прямоугольник
- `Rect(int x, int y, int w, int h)` - с заданными параметрами

Свойства:

- `int x, int y` - координаты верхнего левого угла
- `int width, int height` - размеры

Пример использования:

```
| Rect rect(10, 20, 100, 50); // x=10, y=20, width=100, height=50  
| Rect area = Rect(0, 0, 800, 600);
```

2.2 Point

Точка в 2D-пространстве.

Конструкторы:

- `Point()` - точка (0,0)
- `Point(int x, int y)` - с заданными координатами

Свойства:

- `int x, int y` - координаты

Пример использования:

```
| Point start(0, 0);  
| Point end(100, 100);  
| Point center = Point(50, 50);
```

2.3 Color

Цвет в формате RGBA.

Конструкторы:

- `Color()` - черный непрозрачный
- `Color(int r, int g, int b, int a = 255)` - с компонентами цвета

Свойства:

- `int r, int g, int b` - цветовые компоненты (0-255)
- `int a` - альфа-канал (0-255), где 255 - полностью непрозрачный

Пример использования:

```
| Color red(255, 0, 0);           // Красный  
| Color blue(0, 0, 255, 128);    // Синий полупрозрачный  
| Color white(255, 255, 255);    // Белый
```

2.4 Font

Описание шрифта для текста.

Конструкторы:

- `Font()` - шрифт по умолчанию
- `Font(int size, string name, bool italic = false, bool bold = false, bool underline = false, bool strikethrough = false)`

Свойства:

- `int size` - размер шрифта
- `string name` - название шрифта
- `bool italic` - курсив
- `bool bold` - жирный
- `bool underline` - подчеркнутый
- `bool strikethrough` - зачеркнутый

Пример использования:

```
| Font defaultFont;  
| Font titleFont(24, "Arial", false, true);           // Жирный Arial 24px  
| Font italicFont(12, "Times New Roman", true);      // Курсивный
```

2.5 Pen

Инструмент для рисования линий и контуров.

Конструкторы:

- `Pen()` - перо по умолчанию
- `Pen(int size, Color color, LineType lineType = LineType::Solid)`

Свойства:

- `int size` - толщина линии
- `Color color` - цвет

- `LineStyle lineType` - тип линии

Перечисление `LineStyle`:

- `Undefined` - неопределенный тип
- `Solid` - сплошная линия
- `Dash` - пунктирная линия
- `Dot` - точечная линия
- `DashDot` - штрих-пунктирная
- `DashDotDot` - штрих-две точки

Пример использования:

```
| Pen thinRed(1, Color(255, 0, 0), LineType::Solid);  
| Pen thickBlue(3, Color(0, 0, 255), LineType::Dash);  
| Pen dottedBlack(2, Color(0, 0, 0), LineType::Dot);
```

2.6 ResourceLink

Ссылка на ресурсы (изображения, звуковые файлы).

Конструкторы:

- `ResourceLink()` - пустая ссылка
- `ResourceLink(const string &in path)` - из файлового пути
- `ResourceLink(const string &in lib, const string &in res)` - из библиотеки ресурсов

Методы:

- `string GetFilePath() const` - путь к файлу
- `string GetFullFilePath() const` - полный путь
- `string GetLibraryName() const` - имя библиотеки
- `string GetResourceName() const` - имя ресурса
- `bool IsValid()` - проверка валидности
- `bool IsEmpty()` - проверка на пустоту
- `LinkType GetLinkType() const` - тип ссылки
- `string ToString()` - строковое представление

Перечисление `LinkType`:

- `Undefined` - неопределенный тип
- `Resource` - ресурс из библиотеки
- `File` - файл из файловой системы

Пример использования:

```
| ResourceLink fileImage("images/background.png");  
| ResourceLink libImage("textures", "wood_pattern");  
| ResourceLink invalidLink;  
|  
| if (fileImage.IsValid())  
| {  
|     string path = fileImage.GetFilePath();  
| }  
|
```

2.7 Gradient

Градиентная заливка.

Конструкторы:

- `Gradient()` - градиент по умолчанию
- `Gradient(GradientType type)` - с указанием типа

Свойства:

- `GradientType type` - тип градиента
- `Point start, Point end` - начальная и конечная точки (для линейного)
- `Point center` - центр (для радиального)
- `float radius` - радиус (для радиального)
- `Point focal` - фокальная точка
- `Point centerConical` - центр конического градиента
- `float angle` - угол

Методы для работы с цветовыми остановками:

- `void AddColorStop(float position, Color color)` - добавить цветовую остановку
- `void ClearColorStops()` - очистить все остановки
- `int GetColorStopCount() const` - количество остановок
- `void GetColorStop(int index, float& out position, Color& out color) const` - получить остановку
- `void SetColorStop(int index, float position, Color color)` - установить остановку

Перечисление GradientType:

- `Linear` - линейный градиент
- `Radial` - радиальный градиент
- `Conical` - конический градиент

Пример использования:

```

| Gradient linearGrad(GradientType::Linear);
| linearGrad.start = Point(0, 0);
| linearGrad.end = Point(100, 0);
| linearGrad.AddColorStop(0.0, Color(255, 0, 0)); // Красный в начале
| linearGrad.AddColorStop(1.0, Color(0, 0, 255)); // Синий в конце
|
| Gradient radialGrad(GradientType::Radial);
| radialGrad.center = Point(50, 50);
| radialGrad.radius = 50;

```

2.8 Brush

Инструмент заливки.

Конструкторы:

- `Brush()` - кисть по умолчанию
- `Brush(Color color, BrushStyle style = BrushStyle::SolidPattern)` - сплошной цвет
- `Brush(Gradient gradient, BrushStyle style = BrushStyle::LinearGradientPattern)` - градиент
- `Brush(ResourceLink texture, BrushStyle style = BrushStyle::TexturePattern)` - текстура

Свойства:

- `BrushStyle style` - стиль кисти
- `Color color` - цвет (для `SolidPattern`)
- `Gradient gradient` - градиент
- `ResourceLink texture` - текстура

Перечисление `BrushStyle`:

- **Сплошные и пустые:** `NoBrush`, `SolidPattern`
- **Плотность заливки:** `Dense1Pattern` - `Dense7Pattern` (от самой плотной до самой редкой)
- **Линейные паттерны:** `HorPattern` (горизонтальные линии), `VerPattern` (вертикальные линии), `CrossPattern` (сетка)
- **Диагональные паттерны:** `BDiagPattern`, `FDiagPattern`, `DiagCrossPattern`
- **Специальные:** `LinearGradientPattern`, `RadialGradientPattern`, `ConicalGradientPattern`, `TexturePattern`

Пример использования:

```

| Brush solidRed(Color(255, 0, 0));
| Brush gradientBrush(linearGrad); // Градиентная кисть
| Brush textureBrush(textureLink, BrushStyle::TexturePattern);
| Brush patternBrush(Color(0, 0, 0), BrushStyle::CrossPattern); // Сетка
|
|

```

2.9 Alignment

Флаги выравнивания текста.

Значения:

- **Горизонтальное:** `AlignLeft`, `AlignRight`, `AlignHCenter`, `AlignJustify`, `AlignAbsolute`
- **Вертикальное:** `AlignTop`, `AlignBottom`, `AlignVCenter`, `AlignBaseline`
- **Комбинированные:** `AlignCenter` (центр по горизонтали и вертикали)

Пример использования:

```
| Alignment leftTop = Alignment::AlignLeft | Alignment::AlignTop;  
| Alignment center = Alignment::AlignCenter;  
| Alignment rightBottom = Alignment::AlignRight | Alignment::AlignBottom;
```

3 Класс Painter

Основной класс для рисования, предоставляющий широкий набор графических операций.

3.1 Методы рисования фигур

Дуги и секторы

- `DrawArc(const Rect &in, int startAngle, int spanAngle)` - дуга в прямоугольнике
- `DrawArc(int x, int y, int w, int h, int startAngle, int spanAngle)` - дуга с координатами
- `DrawChord()` - хорда (замкнутая дуга)
- `DrawPie()` - сектор круга

Пример:

```
| painter.DrawArc(Rect(10, 10, 100, 100), 45, 90); // Дуга от 45 до 135 градусов  
| painter.DrawPie(50, 50, 80, 80, 0, 90); // Сектор 90 градусов
```

Эллипсы и окружности

- `DrawEllipse(const Rect &in)` - эллипс в прямоугольнике
- `DrawEllipse(int x, int y, int w, int h)` - эллипс с координатами
- `DrawEllipse(const Point &in center, int rx, int ry)` - эллипс из центра

Пример:

```
| painter.DrawEllipse(Rect(0, 0, 200, 100)); // Эллипс
| painter.DrawEllipse(Point(100, 100), 50, 50); // Окружность
```

Линии и точки

- DrawLine(int x1, int y1, int x2, int y2) - линия между точками
- DrawLine(const Point &in start, const Point &in end) - линия между точками
- DrawPoint(const Point &in) - точка
- DrawPoint(int x, int y) - точка с координатами

Пример:

```
| painter.DrawLine(0, 0, 100, 100); // Диагональная линия
| painter.DrawLine(Point(10, 10), Point(50, 50)); // Линия между точками
| painter.DrawPoint(25, 25); // Точка
```

Прямоугольники

- DrawRect(const Rect &in) - прямоугольник
- DrawRect(int x, int y, int w, int h) - прямоугольник с координатами
- DrawRoundedRect(const Rect &in, double xRadius, double yRadius) - скругленный прямоугольник
- DrawRoundedRect(int x, int y, int w, int h, double xRadius, double yRadius) - скругленный с координатами

Пример:

```
| painter.DrawRect(Rect(10, 10, 100, 50)); // Прямоугольник
| painter.DrawRoundedRect(20, 20, 80, 40, 10, 10); // Скругленный прямоугольник
```

Изображения

- DrawImage(const Rect &in target, const ResourceLink &in image, const Rect &in source) - изображение с областью
- DrawImage(const Point &in pos, const ResourceLink &in image, const Rect &in source) - изображение в позиции с областью
- DrawImage(const Rect &in target, const ResourceLink &in image) - изображение в прямоугольник
- DrawImage(const Point &in pos, const ResourceLink &in image) - изображение в позиции

Пример:

```
| ResourceLink img("images/icon.png");
| painter.DrawImage(Point(0, 0), img); // Рисуем в позиции (0,0)
| painter.DrawImage(Rect(10, 10, 32, 32), img); // Масштабируем в прямоугольник
| painter.DrawImage(Rect(0, 0, 64, 64), img, Rect(0, 0, 32, 32)); // Часть изображения
```

Текст

- `DrawText(const Point &in pos, const string &in text)` - текст в позиции
- `DrawText(int x, int y, const string &in text)` - текст с координатами
- `DrawText(const Rect &in rect, int alignment, const string &in text)` - текст в прямоугольнике с выравниванием
- `DrawText(int x, int y, int w, int h, int alignment, const string &in text)` - текст в области с выравниванием

Пример:

```
| painter.DrawText(Point(10, 10), "Hello World"); // Простой текст  
| painter.DrawText(Rect(0, 0, 200, 50), Alignment::AlignCenter, "Centered Text"); // Выровненный текст
```

3.2 Методы заливки и очистки

- `EraseRect(const Rect &in)` - очистить прямоугольную область
- `EraseRect(int x, int y, int w, int h)` - очистить область с координатами
- `FillRect(const Rect &in, const Brush &in)` - залить прямоугольник кистью
- `FillRect(int x, int y, int w, int h, const Brush &in)` - залить область кистью
- `FillRect(const Rect &in, const Color &in)` - залить прямоугольник цветом
- `FillRect(int x, int y, int w, int h, const Color &in)` - залить область цветом

Пример:

```
| painter.FillRect(Rect(0, 0, 100, 100), Color(255, 0, 0)); // Красный прямоугольник  
| painter.FillRect(10, 10, 50, 50, gradientBrush); // Градиентная заливка  
| painter.EraseRect(5, 5, 10, 10); // Очистить область
```

3.3 Управление состоянием

Геттеры и сеттеры стилей

- `Brush GetBackground(), void SetBackground(const Brush &in)` - фон
- `Brush GetBrush(), void SetBrush(const Brush &in)` - кисть заливки
- `Point GetBrushOrigin(), void SetBrushOrigin(const Point &in)` - начало координат кисти
- `Font GetFont(), void SetFont(const Font &in)` - шрифт текста
- `Pen GetPen(), void SetPen(const Pen &in)` - перо для линий

Пример:

```

| // Установка стилей
| painter.SetPen(Pen(2, Color(0, 0, 0)));
| painter.SetBrush(Brush(Color(255, 255, 0)));
| painter.SetFont(Font(12, "Arial", false, true));
|
| // Получение текущих стилей
| Pen currentPen = painter.GetPen();
| Brush currentBrush = painter.GetBrush();

```

3.4 Трансформации и clipping

Трансформации

- void ResetTransform() - сброс всех трансформаций
- void Rotate(double angle) - поворот на угол (в градусах)
- void Scale(double sx, double sy) - масштабирование
- void Shear(double sh, double sv) - наклон/скос
- void Translate(const Point &in) - смещение
- void Save(), void Restore() - сохранение/восстановление состояния

Clipping (область отсечения)

- bool HasClipping() - проверка наличия области отсечения
- void SetClipRect(const Rect &in, int operation = 1) - установка прямоугольной области отсечения
- void SetClipRect(int x, int y, int w, int h, int operation = 1) - установка области с координатами
- void SetClipping(bool enable) - включить/выключить отсечение

Пример:

```

| // Трансформации
| painter.Save(); // Сохраняем состояние
| painter.Translate(Point(100, 100)); // Смещаем начало координат
| painter.Rotate(45); // Поворачиваем на 45 градусов
| painter.Scale(1.5, 1.0); // Масштабируем
| // Рисуем трансформированные объекты
| painter.Restore(); // Восстанавливаем состояние
|
| // Clipping
| painter.SetClipRect(Rect(50, 50, 100, 100)); // Устанавливаем область отсечения
| // Все рисование будет обрезано по этой области

```

3.5 Вспомогательные методы

- bool HasClipping() - проверка наличия clipping
- int GetLayoutDirection(), void SetLayoutDirection(int) - направление layout (LTR/RTL)
- double GetOpacity(), void SetOpacity(double) - прозрачность (0.0-1.0)

- `Rect GetBoundingRect(const Rect &in, int alignment, const string &in text)` - вычисление ограничивающего прямоугольника для текста
- `Rect GetBoundingRect(int x, int y, int w, int h, int alignment, const string &in text)` - вычисление с координатами

Пример:

```

| // Вычисление размера текста
| Rect textBounds = painter.GetBoundingRect(Rect(0, 0, 0, 0), Alignment::AlignLeft, "Sample Text");
| int textWidth = textBounds.width;
| int textHeight = textBounds.height;
|
| // Установка прозрачности
| painter.SetOpacity(0.5); // Полупрозрачное рисование

```

3.6 Полный пример использования Painter

```

| // Создание инструментов
| Pen borderPen(2, Color(0, 0, 0), LineType::Solid);
| Brush fillBrush(Color(255, 255, 0));
| Font textFont(14, "Arial", false, true);
|
| // Настройка painter
| painter.SetPen(borderPen);
| painter.SetBrush(fillBrush);
| painter.SetFont(textFont);
|
| // Рисование
| painter.DrawRect(Rect(10, 10, 100, 50)); // Прямоугольник
| painter.DrawEllipse(Point(150, 35), 25, 15); // Эллипс
| painter.DrawLine(Point(200, 10), Point(250, 60)); // Линия
| painter.DrawText(Point(15, 15), "Hello World"); // Текст
|
| // Градиентная заливка
| Gradient grad(GradientType::Linear);
| grad.start = Point(0, 0);
| grad.end = Point(100, 0);
| grad.AddColorStop(0.0, Color(255, 0, 0));
| grad.AddColorStop(1.0, Color(0, 0, 255));
| Brush gradBrush(grad);
| painter.SetBrush(gradBrush);
| painter.FillRect(Rect(300, 10, 100, 50), gradBrush);

```

4 Обработка событий

4.1 MouseEvent

Событие мыши.

Методы:

- `MouseButton button()` const - нажатая кнопка
- `Point pos()` const - позиция курсора
- `int x()` const - координата X

- `int y() const` - координата Y

Перечисление `MouseButton`:

- `NoButton` - нет нажатой кнопки
- `LeftButton` - левая кнопка мыши
- `RightButton` - правая кнопка мыши
- `MiddleButton` - средняя кнопка мыши

Пример использования:

```
void OnMouseClicked(MouseEvent@ event)
{
    if (event.button() == MouseButton::LeftButton)
    {
        Point clickPos = event.pos();
        int x = event.x();
        int y = event.y();
    }
}
```

4.2 WheelEvent

Событие колесика мыши.

Методы:

- `int angleDelta() const` - угол поворота колесика (положительный - вверх, отрицательный - вниз)

Пример использования:

```
void OnWheel(WheelEvent@ event)
{
    int delta = event.angleDelta();
    if (delta > 0)
    {
        // Прокрутка вверх
    }
    else
    {
        // Прокрутка вниз
    }
}
```

Источник —

https://docs.kb-agava.ru/index.php?title=Классы_для_рисования_и_работы_с_графическим_интерфейсом_AgavaSCADA/AgavaPLC&oldid=3474