Графический контекст компонентов

Графику в Java обслуживают классы Graphics и Graphics2D. Работа с графикой осуществляется в графическом контексте элементов, унаследованных от класса Component. Понимать это можно так: на элементах управления, например, Frame, Panel, Button и других, есть возможность рисовать. Такие элементы обладают графическим контекстом, в этом контескте мы и рисуем. Всё, что нарисуем в контексте будет показано на элементе. Классы Graphics и Graphics2D нужны для работы с графическим контекстом. Мы должны получить экземпляр такого класса и, используя его методы, рисовать. Получить экземпляр контекста можно в методе paint:

public void paint(Graphics g);

этот метод наследуется из класса Component. Аргумент Graphics g создаётся системой, а мы берём его в готовом виде и используем для рисования. При создании элемента метод paint будет вызван автоматически.

Начнём изучать работу с графикой в Java с класса Graphics.

Graphics

Рассмотрим простой пример использования методов класса Graphics в Java:

Получаем:   
 public void paint(Graphics g)

{

g.drawLine(20, 20, 360, 20);

Color oldColor = g.getColor();

Color newColor = new Color(0, 0, 255);

g.setColor(newColor);

g.drawLine(20, 30, 360, 30);

g.setColor(oldColor);

g.drawRect(20, 40, 340, 20);

newColor = new Color(0, 215, 255);

g.setColor(newColor);

g.fillRect(21, 41, 339, 19);

g.setColor(oldColor);

g.drawRoundRect(20, 70, 340, 30, 20, 15);

g.drawOval(20, 110, 150, 60);

g.drawOval(200, 110, 60, 60);

g.drawArc(280, 110, 80, 60, 0, 180);

int[] arrayX = {20, 100, 100, 250, 250, 20, 20, 50};

int[] arrayY = {180, 180, 200, 200, 220, 200, 200, 190};

Polygon poly = new Polygon(arrayX, arrayY, 8);

g.drawPolygon(poly);

Point aPoint = new Point(50, 190);

if(poly.contains(aPoint))

{

g.drawString("Yes", 50, 190);

}

newColor = new Color(0, 0, 255);

g.setColor(newColor);

Font font = new Font("Tahoma", Font.BOLD|Font.ITALIC, 40);

Font oldFont = g.getFont();

g.setFont(font);

g.drawString("SBP", 270, 220);

g.setFont(oldFont);

g.setColor(oldColor);

// Draw axes;

g.drawLine(20, 220, 20, 350);

g.drawLine(20, 350, 360, 350);

g.drawString("Y", 25, 230);

g.drawString("X", 350, 346);

// Draw a curve;

int[] xArray = {20, 40, 60, 80, 100, 120, 130, 140, 280, 332};

int[] yArray = {350, 345, 340, 310, 290, 280, 275, 273, 271, 269};

int nPoint = 10;

g.setColor(newColor);

g.drawPolyline(xArray, yArray, nPoint);

g.setColor(oldColor);

g.drawString("y = f(x)", 180, 267);

}

Ниже разбираются все методы, использованные в примере.

[Наверх](http://www.sbp-program.ru/java/sbp-graphics.htm)

Как начертить прямую линию?

Метод drawLine класса Graphics начертит прямую линию:

g.drawLine(20, 30, 360, 30);

здесь 20, 30 — это координаты x, y начала линии,  
360, 30 — координаты конца линии.

Как задать цвет?

Метод setColor класса Graphics сделает текущим новый цвет:

// Запоминаем исходный цвет;  
Color oldColor = g.getColor();  
// Создаём синий цвет;  
Color newColor = new Color(0, 0, 255);  
// Устанавливаем синий цвет;  
g.setColor(newColor);  
// Чертим линию синим цветом;  
g.drawLine(20, 30, 360, 30);  
// Восстанавливаем исходный цвет;  
g.setColor(oldColor);

Аргументы конструктора new Color(0, 0, 255) — это красный, зелёный и синий цвета соответственно (rgb).

Как задать rgb цвета? В примере задан чисто синий цвет, т.к. значения других составляющих равны нулю. Вот чисто красный цвет:

Color newColor = new Color(255, 0, 0);

А это чисто зеленый цвет:

Color newColor = new Color(0, 255, 0);

Значения составляющих цвета изменяются от 0 до 255.

Светло-синий цвет, который мы использовали для заливки прямоугольника:

newColor = new Color(0, 215, 255);

[Наверх](http://www.sbp-program.ru/java/sbp-graphics.htm)

Как задать цвет фона?

Задать цвет фона можно методом setBackground:

mainFrame.setBackground(Color.white);

Как нарисовать прямоугольник?

Методом drawRect класса Graphics:

g.drawRect(20, 40, 340, 20);

20, 40 — это координаты верхнего левого угла прямоугольника;  
340 — длина;  
20 — высота прямоугольника.

Как залить прямоугольник цветом?

Методом fillRect класса Graphics:

newColor = new Color(0, 215, 255);  
g.setColor(newColor);  
g.fillRect(21, 41, 339, 19);  
g.setColor(oldColor);

Как нарисовать прямоугольник с закругленными углами?

Методом drawRoundRect класса Graphics.

Сопряжение, т.е. закругление на углах, делается с помощью частей овала.

g.drawRoundRect(20, 70, 340, 30, 20, 15);

первые 4 аргумента как у обычного прямоугольника. Пятый аргумент — 20 — это ширина прямоугольника, в который вписана часть овала сопряжения. Шестой аргумент — 15 — это высота прямоугольника, в который вписана часть овала сопряжения.

[Наверх](http://www.sbp-program.ru/java/sbp-graphics.htm)

Как нарисовать овал?

Методом drawOval класса Graphics:

g.drawOval(20, 110, 150, 60);

Аргументы определяют прямоугольник, в который вписан овал.

Как нарисовать окружность?

Методом drawOval класса Graphics:

g.drawOval(200, 110, 60, 60);

Аргументы определяют прямоугольник, в который вписана окружность. Здесь рисуем овал, но длина и высота описанного прямоугольника равны, что и даёт окружность.

Как нарисовать дугу?

Методом drawArc класса Graphics:

g.drawArc(280, 110, 80, 60, 0, 180);

первые 4 аргумента как у обычного прямоугольника. Пятый аргумент — 0 — это угол, от которого отсчитывается угол самой дуги. 180 — это угол дуги. Углы отсчитывают от горизонтальной оси: по часовой стрелке отрицательное направление, протв — положительное. В примере 180 градусов (величина дуги) отсчитываем от горизонтальной линии.

Как нарисовать многоугольник?

Методом drawPolygon класса Graphics:

int[] arrayX = {20, 100, 100, 250, 250, 20, 20, 50};  
int[] arrayY = {180, 180, 200, 200, 220, 200, 200, 190};  
Polygon poly = new Polygon(arrayX, arrayY, 8);  
g.drawPolygon(poly);

Здесь создаём объект класса Polygon. arrayX — это х-координаты вершин многоугольника, arrayY — это y-координаты вершин многоугольника, 8 — число вершин многоугольника.

[Наверх](http://www.sbp-program.ru/java/sbp-graphics.htm)

Как создать объект точки?

Для этого используем класс Point:

Point aPoint = new Point(50, 190);

аргументы — это x, y координаты.

Как определить, что точка принадлежит многоугольнику?

Polygon poly = new Polygon(arrayX, arrayY, 8);  
g.drawPolygon(poly);  
Point aPoint = new Point(50, 190);  
if(poly.contains(aPoint))  
{  
    g.drawString("Yes", 50, 190);  
}

Используем метод класса Polygon contains для определения лежит ли точка в многоугольнике.

Как вывести строку?

Методом drawString класса Graphics:

g.drawString("Yes", 50, 190);

строка "Yes" будет выведена от точки с координатами 50, 190.

[Наверх](http://www.sbp-program.ru/java/sbp-graphics.htm)

Как задать шрифт?

Для этого используем класс Font:

Font font = new Font("Tahoma", Font.BOLD|Font.ITALIC, 40);

где "Tahoma" — название шрифта,   
Font.BOLD|Font.ITALIC — жирный шрифт с наклоном,   
40 — высота шрифта.

После задания шрифта мы делаем его текущим и выводим строку этим шрифтом:

g.setFont(font);   
g.drawString("SBP", 270, 220);

Как задать цвет текста?

Чтоб задать цвет текста создадим и установим в графический контекст новый цвет:

newColor = new Color(0, 0, 255);   
g.setColor(newColor);

Здесь мы создали чисто синий цвет. А теперь выводим строку синим цветом:

g.drawString("SBP", 270, 220);

Как начертить график?

Как график функции начертить? Сначала начертим координатные оси:

// Draw axes;   
g.drawLine(20, 220, 20, 350);   
g.drawLine(20, 350, 360, 350);   
g.drawString("Y", 25, 230);   
g.drawString("X", 350, 346);

А теперь построить график функции можно просто. Для этого используем метод drawPolyline класса Graphics:

// Draw a curve;   
int[] xArray = {20, 40, 60, 80, 100, 120, 130, 140, 280, 332};   
int[] yArray = {350, 345, 340, 310, 290, 280, 275, 273, 271, 269};   
int nPoint = 10;   
g.setColor(newColor);   
g.drawPolyline(xArray, yArray, nPoint);   
g.setColor(oldColor);   
g.drawString("y = f(x)", 180, 267);

График строим по точкам, xArray — это x-координаты точек, yArray — y-координаты точек графика, nPoint — это число точек.

Наш график являет собой кривую намагничивания. Но почему график такой угловатый (см. картинку выше)? Если взять больше точек, то график будет более плавным.