# Основы компьютерной графики

Лекция 1

# Задачи компьютерной графики

- Компьютерная обработка графической информации сводится к решению 3-х классов задач:
  - распознавание образа ***изображение описание объекта***
  - обработка изображения с выполнением преобразования ***изображение изображение***
  - построение изображения с выполнением преобразования ***описание объекта изображение.***

# Задача компьютерной геометрии

- Последняя задача и является предметом нашего рассмотрения
- Описание объекта должно иметь вид математической модели

# Подзадачи компьютерной геометрии

- В рамках компьютерной геометрии решаются две основные подзадачи:
  - построение математической модели изображаемого объекта;
  - визуализация объекта в соответствии с этой моделью.

# Информационное содержание изображения

- Определительная информация – *идентификация* и *структура*
- Топологическая информация – *морфология* и *геометрия*
- Визуальная информация – *внешний вид* и *освещение*

# Определительная информация

- *Идентификация* основана на именовании объектов или множеств объектов
- *Структура* отражает различные виды отношений объектов между собой:
  - логические отношения (принадлежность, включение);
  - топологические отношения (близость, касание);
  - функциональные отношения (зависимость характеристик).

# Топологическая информация

- *Морфология* отражает форму объекта независимо от его положения и точки наблюдения. Многообразие всех геометрических объектов является комбинацией различных примитивов
- *Геометрия* отражает информацию о проекциях, видимости и т.д.

# Визуальная информация

- *Внешний вид* определяется свойствами материала поверхности объекта - его цветом, текстурой, прозрачностью и пр.
- *Освещение* определяется природой, числом и расположением источников света, а также условиями видимости: наличием дыма, тумана и т.д.

# Графические средства платформы .Net Framework

- В числе библиотек классов платформы .Net Framework имеется графическая библиотека System.Drawing.dll
- Эта библиотека определяет несколько пространств имен, вложенных в System:
  - Drawing
    - Drawing2D
    - Printing
    - Imaging
  - DrawingText

# Пространства имен графических классов

| Пространство имен | Назначение |
|---|---|
| Drawing | Определяет типы для базовой визуализации (шрифты, перья, базовые кисти и т.п.), а также тип Graphics |
| Drawing.Drawing2D | Представляет типы, используемые для развитой двумерной векторной графики (градиентные кисти, концы перьев, геометрические трансформации и т.п.) |
| Drawing.Printing | Определяет типы, позволяющие печатать на бумаге, взаимодействовать с принтером и форматировать общий вид задания печати |
| Drawing.Imaging | Определяет типы, позволяющие манипулировать графическими образами (изменять палитры, извлекать метаданные изображений и т.п.) |
| DrawingText | Позволяет манипулировать коллекциями шрифтов |

# Пространство имен System.Drawing

- Содержит большинство графических классов и других типов
- Здесь есть классы, представляющие изображения, кисти, перья и шрифты
- Кроме того, *System.Drawing* определяет множество служебных типов, таких как *Color*, *Point*, *Size* и *Rectangle*

# Класс Graphics

- Класс Graphics представляет поверхность рисования
- Есть три основных типа поверхностей рисования:
  - форма и некоторые из управляющих элементов;
  - страницы, посылаемые на принтер;
  - участки в оперативной памяти, выделяемые для построения растровых изображений

# Графический объект

- Создание любого GDI-проекта начинается с создания экземпляра класса Graphics – *графического объекта*, который содержит методы построения геометрических примитивов
- Графический объект поддерживает состояние поверхности рисования, включая масштаб и единицы, так же как ориентацию поверхности рисования

# Создание графического объекта

- Для создания графического объекта вызывается метод CreateGraphics() класса, соответствующего элемента – формы или элемента управления, помещенного на форму
- После использования графический элемент должен быть удален методом Dispose()
- Пример – проект «Начала графики»

# Двойная буферизация

- В тех случаях, когда изображение является сложным, его построение непосредственно на форме может потребовать достаточно много времени
- Решение: строить изображение в памяти, а затем переносить его на форму – эта технология называется *двойной буферизацией*
- Такое предварительно сохраненное в памяти изображение называется *образом*

# Двойная буферизация

- Существует несколько способов получения образа:
  - загрузка из файла;
  - создание из уже существующего образа;
  - создание пустого образа, в котором потом будет осуществляться рисование

# Класс Image

- Для работы с образами используются наследники абстрактного класса Image – классы Bitmap и Metafile
- Класс Bitmap предназначен для работы с растровыми изображениями, представленными в виде матрицы пикселей
- Образ для экземпляра этого класса может быть считан из файлов форматов .gif, .jpeg, .bmp (проект «Картинка»)

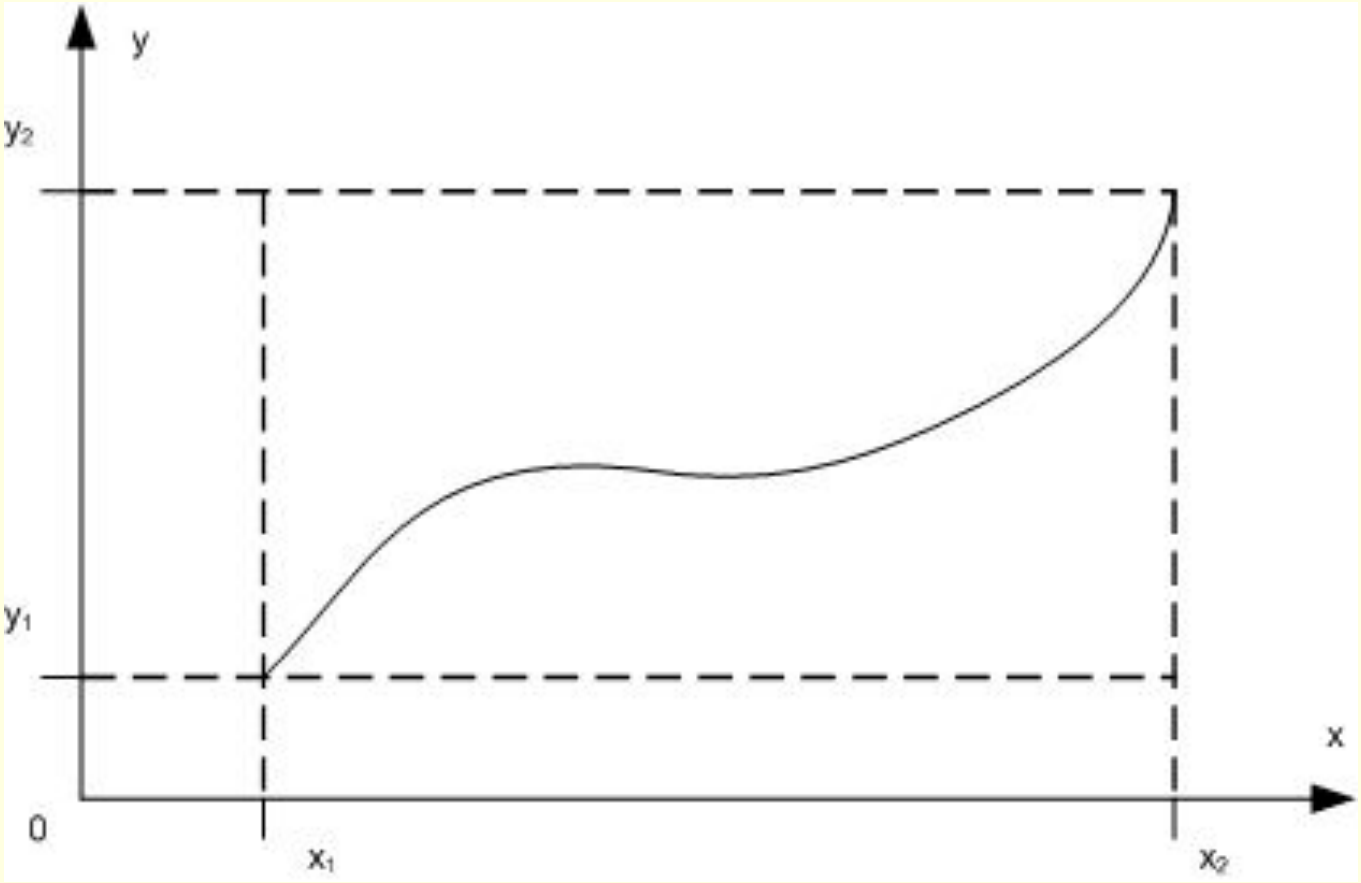
# Класс Image

- Класс Metafile предназначен для работы с векторными изображениями, представленными в виде математического описания образующих его примитивов
- Образ для экземпляра этого класса может быть считан из файлов форматов .wmf и .emf
- Перед выводом на экран векторное изображение предварительно растеризуется

# Построение растровых изображений

- Обычные изображения реализуются на плоскости, являющейся бесконечным и непрерывным множеством точек; положение каждой точки задается парой вещественных чисел (x, y)
- Эти числа рассматриваются как координаты в некоторой системе, которую принято называть *бумажной системой координат*

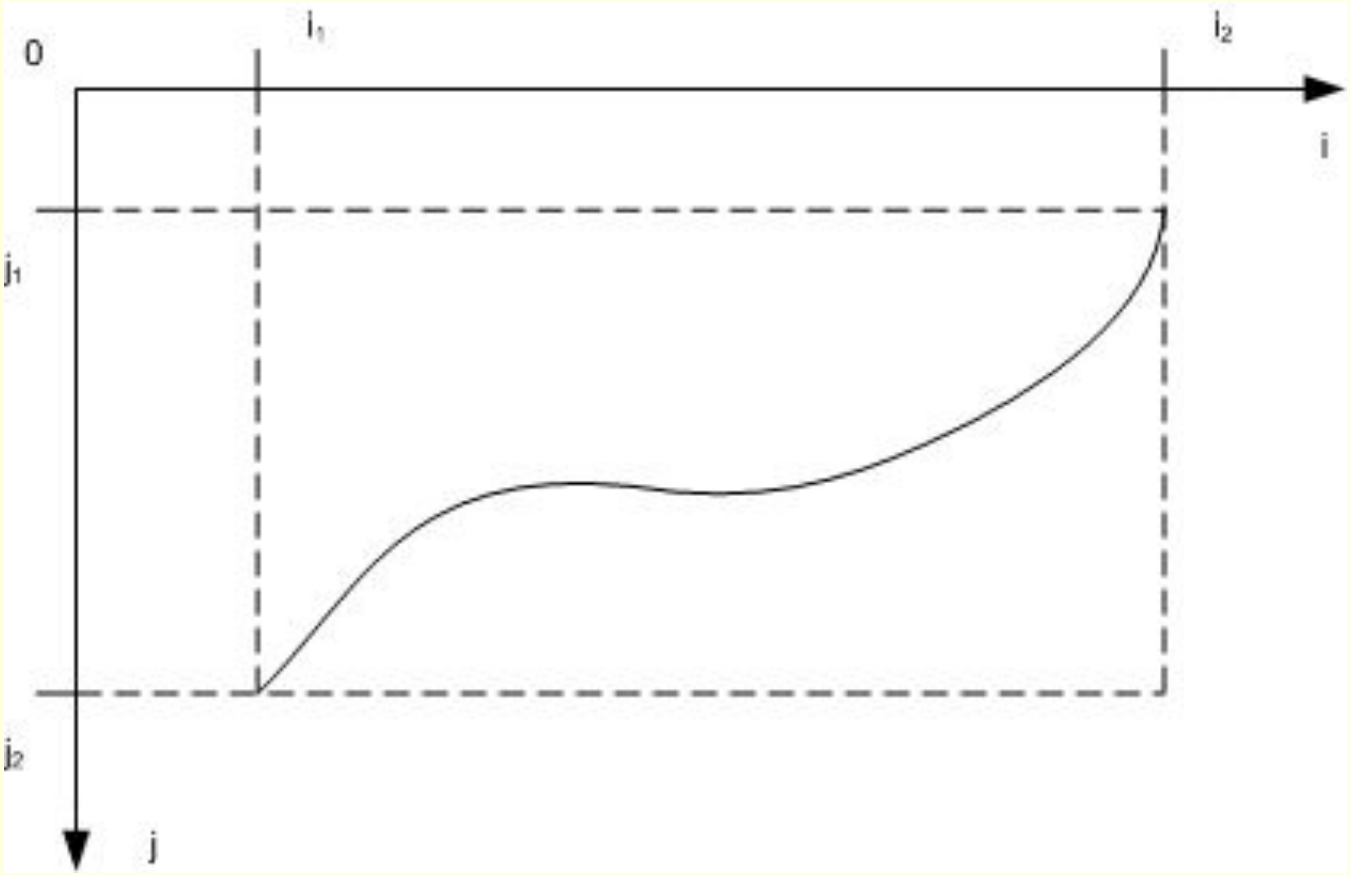
# Бумажная система координат



# Построение растровых изображений

- Графические устройства реализуют изображение в виде *растра* – конечного дискретного набора элементов изображения, называемых *пикселями*
- Пиксели образуют прямоугольную матрицу, в которой положение каждого пикселя задается парой целых чисел (i, j)
- Эти числа естественно рассматривать как координаты точки в *экранной системе координат*

# Экранная система координат



# Формулы преобразования

- Преобразование из бумажных в экранные координаты:

$$i = i_1 + Round\left(\left(x - x_1\right) * \left(i_2 - i_1\right) / \left(x_2 - x_1\right)\right)$$

$$j = j_2 - Round\left(\left(y - y_1\right) * \left(j_2 - j_1\right) / \left(y_2 - y_1\right)\right)$$

# Формулы преобразования

- Преобразование из экранных в бумажные координаты:

$$x = x_1 + (i - i_1) * (x_2 - x_1) / (i_2 - i_1)$$

$$y = y_1 + (j_2 - j) * (y_2 - y_1) / (j_2 - j_1)$$

# Графические методы

- Класс Graphics имеет методы для построения геометрических примитивов
- Названия методов прорисовки линий начинаются с префикса *Draw*
- Названия методов прорисовки закрашенных областей начинаются с префикса *Fill*
- *Draw*-методы рисуют контур фигуры; *Fill*-методы заполняют фигуры заданным цветом

# Методы рисования линий

| Название метода | Описание метода |
|---|---|
| DrawArc | Рисует дугу эллипса |
| DrawBezier | Рисует кривую Безье |
| DrawBeziers | Рисует последовательность кривых Безье |
| DrawClosedCurve | Рисует замкнутую кривую по набору ее точек |
| DrawCurve | Рисует кривую по набору ее точек |
| DrawEllipse | Рисует эллипс |
| DrawLine | Рисует отрезок прямой |
| DrawLines | Рисует ломаную по набору ее точек |
| DrawPie | Рисует сектор эллипса |
| DrawPolygon | Рисует многоугольник |
| DrawRectangle | Рисует прямоугольник |
| DrawRectangles | Рисует набор прямоугольников |

# Методы рисования областей

| Название метода | Описание метода |
|---|---|
| FillClosedCurve | Закрашивает внутреннюю часть замкнутой кривой заданным цветом |
| FillEllipse | Закрашивает внутреннюю часть эллипса заданным цветом |
| FillPath | Закрашивает внутреннюю часть области заданным цветом |
| FillPie | Закрашивает внутреннюю часть сектора эллипса заданным цветом |
| FillPolygon | Закрашивает внутреннюю часть многоугольника заданным цветом |
| FillRectangle | Закрашивает внутреннюю часть прямоугольника заданным цветом |
| FillRectangles | Закрашивает внутреннюю часть набора прямоугольников заданным цветом |

# Инструменты рисования

- Инструментами рисования называются объекты классов *Color, Pen, Brush* и *Font*
- Экземпляр класса Pen называется *пером* и используется для рисования линий и сложных фигур
- *Кисть* – это экземпляр класса Brush, используемый для заполнения формы или рисования текста

# Инструменты рисования

- Объекты Color являются экземплярами классов, которые представляют определенные цвета и могут использоваться перьями и кистями для указания цвета
- Объекты класса Font используются для построения надписей

# Инструмент Перо

- Существует два класса этого инструмента:
  - класс Pen – используется для рисования линий любого цвета;
  - класс Pens – используется для рисования линий стандартных цветов
- Класс Pen имеет 4 конструктора:
  - Pen(Brush brush);
  - Pen(Color color);
  - Pen(Brush brush, float width);
  - Pen(Color color, float width).

# Свойства класса Pen

- Перо обладает следующими важными свойствами:
  - Pen.Brush – получает или задает объект Brush, связанный с объектом Pen;
  - Pen.Color – получает или задает цвет объекта Pen;
  - Pen.Width – получает или задает ширину пера Pen, в единицах объекта;

# Свойства класса Pen

- Pen.PenType – получает или задает стиль линий, нарисованных с помощью объекта Pen;
- Pen.DashStyle – задает стиль пунктирных линий.

# Свойство PenType

- Свойство перечислимого типа, задающее способ заполнения, используемый объектом Pen для заполнения:
  - SolidColor – задает сплошное заполнение;
  - HatchFill – задает заполнение штриховкой;
  - TextureFill – задает заполнение текстурой;
  - LinearGradient – задает линейное градиентное заполнение

# Свойство DashStyle

- Свойство перечислимого типа может принимать следующие значения:
  - Solid – задает сплошную линию.
  - Dash – задает линию, состоящую из штрихов.
  - Dot – задает линию, состоящую из точек.
  - DashDot – адает штрих-пунктирную линию.
  - DashDotDot – задает линию, состоящую из повторяющегося шаблона "штрих-две точки".

# Инструмент Кисть

- Для заливки замкнутых областей используются  наследники абстрактного класса Brush:
  - SolidBrush – простейшая форма кисти, использующая сплошной цвет краски;
  - HatchBrush – аналогична SolidBrush, но позволяющая выбрать из большого разнообразия представленных шаблонов, а не сплошной цвет;

# Инструмент Кисть

- TextureBrush – инициализирует новый объект, использующий указанное изображение и ограничивающий прямоугольник;
- LinearGradientBrush – рисует двумя цветами, смешанными вдоль градиента;
- PathGradientBrush – краски с использованием сложного градиента смешанных цветов, на основе уникального пути, определяемые разработчиком

# Конец лекции