Курс “Компьютерная графика”

1. **Цель курса**
2. **Организация занятий**
3. **Части курса**
   - **вводный курс;**
   - **основные алгоритмы;**
   - **архитектуры графических систем.**
4. **Результаты и контроль**
5. **Литература**

Основные алгоритмы КГ

11 Тем

4 темы, пройденные в предыдущем семестре, подчеркнуты

1. КООРДИНАТЫ И ПРЕОБРАЗОВАНИЯ

- Двумерные преобразования
- Двумерные преобразования в однородных координатах
- Композиция двумерных преобразований
- Эффективность преобразований
- Трехмерные координаты
- Проекции
- Стереоизображения
- Геометрические преобразования растровых картин

2. ГЕНЕРАЦИЯ ВЕКТОРОВ

- Цифровой дифференциальный анализатор
- Алгоритм Брезенхема
- Улучшение качества аппроксимации векторов
- Улучшение качества изображения фильтрацией

3. ГЕНЕРАЦИЯ ОКРУЖНОСТИ

- Алгоритм Брезенхема
- Модифицированный алгоритм Брезенхема

4. ЗАПОЛНЕНИЕ МНОГОУГОЛЬНИКА

- Построчное заполнение
- Сортировка методом распределяющего подсчета

5. ЗАЛИВКА ОБЛАСТИ С ЗАТРАВКОЙ

- Простой алгоритм заливки
- Построчный алгоритм заливки с затравкой

6. ОТСЕЧЕНИЕ ОТРЕЗКОВ

- Двумерный алгоритм Коэна-Сазерленда
- Двумерный FC-алгоритм
- Двумерный алгоритм Лианга-Барски
- Двумерный алгоритм Кируса-Бека
- Сравнение алгоритмов двумерного отсечения
- Трехмерное отсечение отрезка
- Отсечение отрезка в однородных координатах

7. ОТСЕЧЕНИЕ МНОГОУГОЛЬНИКА

- Алгоритм Сазерленда-Ходгмана
- Простой алгоритм отсечения многоугольника
- Алгоритм отсечения многоугольника Вейлера-Азертона

8. СТРУКТУРЫ ДАННЫХ

- Последовательный доступ
- Непосредственный доступ
- Линейные списки
- Комбинированные списки
- Циклические списки

9. ГЕОМЕТРИЧЕСКОЕ МОДЕЛИРОВАНИЕ

- Элементы моделей
- Методы построения моделей
- Типы моделей
- Полигональные сетки
- Внутреннее представление моделей

10. УДАЛЕНИЕ СКРЫТЫХ ЛИНИЙ И ПОВЕРХНОСТЕЙ

- Классификация методов удаления невидимых частей
- Алгоритмы удаления линий
- Алгоритм удаления поверхностей с Z-буфером
- Построчный алгоритм с Z-буфером
- Алгоритм разбиения области Варнока
- Построчный алгоритм Уоткинса
- Алгоритм трассировки лучей

11. РЕАЛИСТИЧНОЕ ПРЕДСТАВЛЕНИЕ СЦЕН

- **Модели освещения**
- **Механизм диффузного и зеркального отражения света**
- **Модели закраски**
- **Прозрачность**
- **Тени**
- **Фактура**
- **Трассировка лучей**
- **Излучательность**

ПРИЛОЖЕНИЯ

1. **Процедуры преобразований**
2. **Процедуры генерации отрезков**
3. **Процедуры фильтрации**
4. **Процедуры генерации окружности**
5. **Процедуры заполнения многоугольника**
6. **Процедуры заливки области**
7. **Процедуры отсечения отрезка**
8. **Процедуры отсечения многоугольника**

ЛИТЕРАТУРА

1. Вельтмандер П.В. Введение в машинную графику: Учеб. пособие/ Новосиб. ун-т. Новосибирск, 1995. 77 с.
2. Вельтмандер П.В. Машинная графика. Вводный курс:Учеб. пособие в электронном виде.
3. Вельтмандер П.В. Машинная графика. Основные алгоритмы: Учеб. пособие в электронном виде.
4. Вельтмандер П.В. Машинная графика. Введение в архитектуры графических систем: Учеб. пособие в электронном виде.
5. Роджерс Д. Алгоритмические основы машинной графики. Пер. с англ. М.: Мир,1989. 512 с.
6. Павлидис Т. Алгоритмы машинной графики и обработки изображений. Пер. с англ. М.: Радио и связь, 1986.
7. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х книгах. Пер. с англ. М.: Мир, 1985.
8. Гилой В. Интерактивная машинная графика. Пер. с англ. М.: Мир, 1981.
9. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики. Пер. с англ. М.: Мир, 1976.
10. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. М.: “ДИАЛОГ–МИФИ”, 1995. 228 с.
11. Donald Hearn, M. Pauline Baker. Computer Graphics. Prentice Hall, 1994. 652 p.

ДИРЕКТОРИИ

1. **WELT/HARD — вводный курс**
2. **WELT/ALG — основные алгоритмы**
3. **WELT/ARCH — архитектуры графических систем**
4. **WELT/SLIDES — слайды к лекциям**
5. **WELT/KURS — темы заданий**
6. **WELT/EXAM — вопросы к экзамену**

- двумерные (2D) преобразования,
- 2D преобразования в однородных координатах,
- композиция 2D преобразований,
- 3D координаты,
- проекции,
- стереоизображения,
- преобразования растровых картин.

Всюду далее: X,Y,Z — декартовые координаты,
x,y,z — однородные координаты,

Двумерные преобразования

1. Сдвиг:

Скалярная форма

$$Xn = X + Tx,$$
$$Yn = Y + Ty.$$

Векторная форма

$$\vec{Pn} = \vec{P} + \vec{T}.$$

где $\vec{Pn} = [Xn, Yn]$, $\vec{P} = [X, Y]$, $\vec{T} = [Tx, Ty]$.

2. Масштабирование относительно начала координат:

Скалярная форма

$$Xn = X \cdot Sx,$$
$$Yn = Y \cdot Sy.$$

Матричная форма

$$\vec{Pn} = \vec{P} \cdot S,$$
$$[Xn, Yn] = [X, Y] \cdot \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

где $S = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$ — матрица масштабирования.

3. Поворот относительно начала координат:

Скалярная форма	Матричная форма
$\mathbf{Xn} = \mathbf{X} \cdot \cos\phi - \mathbf{Y} \cdot \sin\phi,$	$\vec{\mathbf{Pn}} = \vec{\mathbf{P}} \cdot \mathbf{R}.$
$\mathbf{Yn} = \mathbf{X} \cdot \sin\phi + \mathbf{Y} \cdot \cos\phi,$	

где ϕ — угол поворота,

$$\mathbf{R} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} \mathbf{R_{11}} & \mathbf{R_{12}} \\ \mathbf{R_{21}} & \mathbf{R_{22}} \end{bmatrix} \text{ — матрица поворота,}$$

Столбцы и строки матрицы поворота — взаимно перпендикулярные единичные векторы,
Т.е. квадраты длин векторов строк и столбцов = 1:

$$\begin{aligned}
\mathbf{R_{11}^2} + \mathbf{R_{12}^2} &= 1 = \cos\phi \cdot \cos\phi + \sin\phi \cdot \sin\phi, \\
\mathbf{R_{21}^2} + \mathbf{R_{22}^2} &= 1 = (-\sin\phi) \cdot (-\sin\phi) + \cos\phi \cdot \cos\phi, \\
\mathbf{R_{11}^2} + \mathbf{R_{21}^2} &= 1 = \cos\phi \cdot \cos\phi + (-\sin\phi) \cdot (-\sin\phi), \\
\mathbf{R_{12}^2} + \mathbf{R_{32}^2} &= 1 = \sin\phi \cdot \sin\phi + \cos\phi \cdot \cos\phi,
\end{aligned}$$

а их скалярное произведение = 0:

$$\begin{aligned}
\mathbf{R_{11} \cdot R_{21} + R_{12} \cdot R_{22}} &= 0 = \cos\phi \cdot (-\sin\phi) + \sin\phi \cdot \cos\phi, \\
\mathbf{R_{11} \cdot R_{12} + R_{21} \cdot R_{22}} &= 0 = \cos\phi \cdot \sin\phi + (-\sin\phi) \cdot \cos\phi.
\end{aligned}$$

Кроме того вектора-столбцы после выполнения преобразования, заданного этой матрицей, совпадут с осями.

$$\begin{bmatrix} \cos\phi & -\sin\phi \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{X}$$

$$\begin{bmatrix} \sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad \mathbf{Y}.$$

Это позволяет сформировать матрицу преобразования, если известны его результаты.

2D преобразования в однородных координатах

Формирование суммарного 2D преобразования затруднено, т.к. различные преобразования имеют различные виды. Для устранения затруднений вводятся 2D однородные координаты:

$[\,x\ y\ w\,];\quad w \neq 0 \quad \Longrightarrow X = x/w;\ Y = y/w.$

Однородные координаты можно представить как декартовые, промасштабированные коэффициентом w, расположенные в плоскости с $Z = w$.

Все 2D преобразования в однородных координатах имеют одинаковую форму.

$$\begin{bmatrix} xn & yn & wn \end{bmatrix} = \begin{bmatrix} x & y & w \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix} \qquad \text{сдвиг}$$

$$\begin{bmatrix} xn & yn & wn \end{bmatrix} = \begin{bmatrix} x & y & w \end{bmatrix} \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{масштабирование.}$$

$$\begin{bmatrix} xn & yn & wn \end{bmatrix} = \begin{bmatrix} x & y & w \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{поворот}$$

Видно, что $wn = w$, а матрица преобразования для двумерных однородных координат имеет вид:

$$\left[\begin{array}{cc|c} A & B & P \\ D & E & Q \\ \hline L & M & S \end{array}\right], \quad \begin{bmatrix} A & B \\ D & E \end{bmatrix}$$

определяет масштабирование, поворот, смещение

[L M] определяет сдвиг, S определяет общий масштаб,

$\begin{bmatrix} P \\ Q \end{bmatrix}$ определяет проецирование,

Для уяснения смысла элемента S рассмотрим

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S \end{bmatrix} = \begin{bmatrix} x & y & S \end{bmatrix}$$

Таким образом двумерные декартовые координаты преобразованной точки

$$X = x/S, \qquad Y = y/S,$$

т.е. это преобразование задает изменение масштаба.

Для уяснения смысла вектора [P Q] выполним преобразование

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & P \\ 0 & 1 & Q \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & (Px + Qy + 1) \end{bmatrix} = \begin{bmatrix} xn & yn & h \end{bmatrix},$$

h — переменная, определяющая положение плоскости с преобразованной точкой, из 1 преобразовалась в уравнение 3D плоскости:

$$h = Px + Qy + 1.$$

Двумерные декартовые координаты Xn, Yn для преобразованной точки

$$Xn = \frac{x}{Px + Qy + 1}, \qquad Yn = \frac{y}{Px + Qy + 1}.$$

Это соответствует вычислению их в плоскости $Z = 1$, т.е. проецированию из плоскости $Px + Qy + 1$ в плоскость $Z = 1$.

Покажем, что центр проецирования находится в начале координат.

Рассмотрим для этого параметрические уравнения прямой, проходящей через точки $(Xn,\ Yn,\ 1)$ и $(x,\ y,\ (Px + Qy + 1))$:

$$\begin{aligned} X(t) &= Xn + (x - Xn) \cdot t &&= x/h + (x - x/h) \cdot t, \\ Y(t) &= Yn + (y - Yn) \cdot t &&= y/h + (y - y/h) \cdot t, \\ Z(t) &= 1 + (Px + Qy) \cdot t &&= 1 + (h - 1). \end{aligned}$$

Полагая $X(t) = 0$, получим $t = 1/(1 - h)$. Подставляя это значение в выражения для $Y(t)$ и $Z(t)$, получим:

$$Y(t) = y/h + (y - y/h)/(1 - h) = y/h - y/h = 0.$$

$$Z(t) = 1 + (h - 1)/(1 - h) = 0.$$

Т.е. показано, что вектор [P Q] определяют проецирование с центром проекции в начале координат.

Декартовые точки с бесконечными координатами

Пусть имеем линию, проходящую через декартовую точку (X,Y).

Ее однородные координаты — (x,y,h) = (Xh,Yh,h).

При $h \to 0$ $\lim(x/y) = X/Y$, $x \to \infty$, $y \to \infty$, т.е. (x,y,0) задает в декартовой системе координат точку на ∞ для рассмотренной прямой.

Параллельные прямые

Прямые, параллельные в декартовой системе координат, в однородных координатах имеют точку пересечения.

Пусть две пересекающиеся прямые в декартовой системе координат заданы системой уравнений:

$$\begin{aligned} A_1 \cdot X + B_1 \cdot Y + C_1 &= 0 \\ A_2 \cdot X + B_2 \cdot Y + C_2 &= 0 \end{aligned}$$

Решая эту систему относительно X и Y, найдем координаты точки пересечения

$$\begin{aligned} X_0 &= (C_1 \cdot B_2 - C_2 \cdot B_1)/\Delta, \\ Y_0 &= (A_1 \cdot C_2 - A_2 \cdot C_1)/\Delta, \\ \Delta &= A_1 \cdot B_2 - A_2 \cdot B_1. \end{aligned}$$

Запишем результат в однородных координатах

$$\left(\frac{C_1 \cdot B_2 - C_2 \cdot B_1}{\Delta}, \frac{A_1 \cdot C_2 - A_2 \cdot C_1}{\Delta}, 1 \right).$$

В силу произвольности масштабного множителя, умножим значения координат на Δ

$$(C_1 \cdot B_2 - C_2 \cdot B_1, \;\; A_1 \cdot C_2 - A_2 \cdot C_1, \;\; \Delta).$$

Если прямые параллельны, то определитель системы — Δ равен нулю. Учитывая это и обозначая $x_0 = (C_1 \cdot B_2 - C_2 \cdot B_1)$, $y_0 = (A_1 \cdot C_2 - A_2 \cdot C_1)$, получим координату пересечения параллельных прямых в однородной системе координат

$$(x_0, y_0, 0).$$

При этом точка пересечения лежит на ∞ на прямой:

$$-y_0 \cdot x + x_0 \cdot y = 0$$

Композиция двумерных преобразований

Последовательное выполнение нескольких преобразований можно представить в виде единой матрицы суммарного преобразования.

Последовательное выполнение сдвигов

Сдвинем точку P_1 на расстояние (Tx_1, Ty_1) в точку P_2:

$$P_2 = P_1 \cdot T_1.$$

Затем сдвинем точку P_2 на расстояние (Tx_2, Ty_2) в точку P_3:

$$P_3 = P_2 \cdot T_2 = (P_1 \cdot T_1) \cdot T_2) = P_1 \cdot (T_1 \cdot T_2) = P_1 \cdot T.$$

Сдвиг аддитивен, т.е. последовательное выполнение T_1 и T_2 д.б. эквивалентно одному сдвигу на расстояние $(Tx_1 + Tx_2, Ty_1 + Ty_2)$.

Для доказательства этого рассмотрим произведение матриц сдвига T_1 и T_2, равное

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx_1 & Ty_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx_2 & Ty_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx_1 + Tx_2 & Ty_1 + Ty_2 & 1 \end{bmatrix}.$$

Итак, получили, что результирующий сдвиг есть $(Tx_1 + Tx_2, Ty_1 + Ty_2)$, т.е. суммарный сдвиг, вычисленный как произведение матриц, как и ожидалось, аддитивен.

Последовательное выполнение масштабирований

Первое масштабирование с коэффициентами (Sx_1, Sy_1), второе с коэффициентами (Sx_2, Sy_2).

Следует ожидать, что суммарное масштабирование будет мультипликативным. Обозначая через S_1 и S_2 матрицы масштабирования, получим:

$$P_1 = P_0 \cdot S_1, P_2 = P_1 \cdot S_2 = (P_0 \cdot S_1) \cdot S_2 = P_0 \cdot (S_1 \cdot S_2) = P_0 \cdot S.$$

Найдем значения элементов матрицы S

$$S = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Итак, получили, что результирующее масштабирование мультипликативно.

Последовательное выполнение поворотов

Можно показать, что два последовательных поворота аддитивны.

Поворот на угол ϕ относительно заданной точки $P(X, Y)$

1. Перенос начала координат в точку $P(X, Y)$.
2. Поворот на угол ϕ вокруг начала координат.
3. Обратный перенос начала координат:

$$Pn = P \cdot T(-X, -Y) \cdot R(\phi) \cdot T(X, Y).$$

В матричной записи:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X & -Y & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X & Y & 1 \end{bmatrix}.$$

Эффективность преобразований

Двумерное преобразование в однородных координатах имеет вид:

$$\mathbf{Pn\,(xn, yn, wn) = P \cdot T = [x, y, w] \cdot \begin{bmatrix} A & B & 0 \\ D & E & 0 \\ L & M & 1 \end{bmatrix}.}$$

где $\begin{bmatrix} A & B \\ D & E \end{bmatrix}$ — объединенная матрица масштабирования и поворота,

а $[\,L \;\; M\,]$ — вектор сдвига.

Непосредственное использованием выражения $P \cdot T$ требует 9 операций умножения и 6 операций сложения.

Но так как значение w может быть выбрано равным 1, а третий столбец T содержит единственный ненулевой элемент, равный 1, то экономное преобразование может быть представлено в виде:

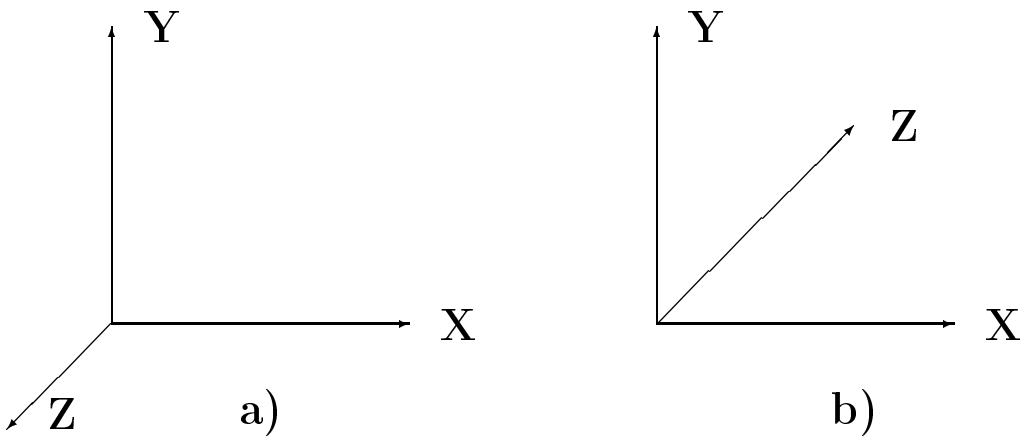
$$Xn = X \cdot A + Y \cdot D + L, \qquad Yn = X \cdot B + Y \cdot E + M,$$

что требует уже только 4 операции умножения и 4 операции сложения.

Таким образом преобразование координат следует производить с учетом реальной структуры матрицы преобразования.

Матрицы же 3×3 удобны при вычислении суммарного преобразования.

Правая и левая системы координат



Система координат называется правой, если для совмещения с положительной полуосью Y положительную полуось X требуется повернуть на $+90^\circ$ при этом направление движения расположенного вдоль оси Z и поворачивающегося против часовой стрелки правого винта и положительной полуоси Z совпадают.

В однородных координатах точка представляется четырехмерным вектором $[\,x\ y\ z\ w\,]$, где $w \neq 0$, а матрицы преобразований имеют размер 4×4.

Переход от однородных к декартовым координатам:

$$\left[\,X\ Y\ Z\ 1\,\right] = \left[\,\frac{x}{w}\ \frac{y}{w}\ \frac{z}{w}\ 1\,\right].$$

Преобразование в однородных координатах:

$$[\,xn\ yn\ zn\ wn\,] = [\,x\ y\ z\ w\,] \cdot T.$$

Матрица преобразования T имеет вид:

$$\left[\begin{array}{ccc|c} A & B & C & P \\ D & E & F & Q \\ I & J & K & R \\ \hline L & M & N & S \end{array}\right],$$

Подматрица 3×3 определяет суммарные смещение, масштабирование и поворот. Вектор $[\,L\,M\,N\,]$ задает сдвиг. Вектор $[\,P\,Q\,R\,]$ отвечает за преобразование в перспективе. Скалярный элемент S определяет общее изменение масштаба.

Матрица сдвига имеет вид:

$$T(Tx, Ty, Tz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}.$$

Матрица обратного преобразования для сдвига получается путем смены знака у Tx, Ty и Tz.

Матрица масштабирования относительно центра координат имеет вид:

$$S(Sx, Sy, Sz) = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица обратного преобразования для масштабирования формируется при замене Sx, Sy и Sz на величины, обратные к ним.

Поворот вокруг оси Z (в плоскости XY)

Размеры вдоль оси Z неизменны, поэтому все элементы 3-й строки и 3-го столбца равны 0, кроме диагонального, равного 1:

$$\mathbf{Rz}(\phi_\mathbf{z}) = \begin{bmatrix} \cos\phi_z & \sin\phi_z & 0 & 0 \\ -\sin\phi_z & \cos\phi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Поворот вокруг оси X (в плоскости YZ)

Размеры вдоль оси X не меняются, поэтому все элементы 1-й строки и 1-го столбца равны 0, за исключением диагонального, равного 1:

$$\mathbf{Rx}(\phi_\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi_x & \sin\phi_x & 0 \\ 0 & -\sin\phi_x & \cos\phi_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Поворот вокруг оси Y (в плоскости XZ)

Размеры вдоль оси Y не меняются, поэтому все элементы 2-й строки и 2-го столбца равны 0, за исключением диагонального, равного 1:

$$\mathbf{Ry}(\phi_\mathbf{y}) = \begin{bmatrix} \cos\phi_y & 0 & -\sin\phi_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi_y & 0 & \cos\phi_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Столбцы и строки подматриц 3×3 матриц поворота **Rx, Ry, Rz** представляют собой взаимно ортогональные единичные векторы.

Суммарная матрица преобразования для произвольной последовательности поворотов вокруг осей X, Y и Z имеет вид:

$$\mathbf{R} = \begin{bmatrix} \mathbf{r1_x} & \mathbf{r1_y} & \mathbf{r1_z} & \mathbf{0} \\ \mathbf{r2_x} & \mathbf{r2_y} & \mathbf{r2_z} & \mathbf{0} \\ \mathbf{r3_x} & \mathbf{r3_y} & \mathbf{r3_z} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Векторы-столбцы при повороте, задаваемом матрицей, совмещаются с соответствующими осями координат.

Матрица, столбцы (или строки) которой представляют собой взаимно ортогональные векторы, называется ортогональной.

Для любой ортогональной матрицы M обратная матрица совпадает с транспонированной. Это обеспечивает простоту вычисления обратного преобразования для поворота. Достаточно просто поменять местами индексы строк и столбцов.

Взаимная ортогональность столбцов матрицы поворота и их совмещение с осями координат при преобразовании, задаваемом матрицей, позволяет легко сконструировать матрицу преобразования, если известны его результаты.

Формирование матрицы преобразования

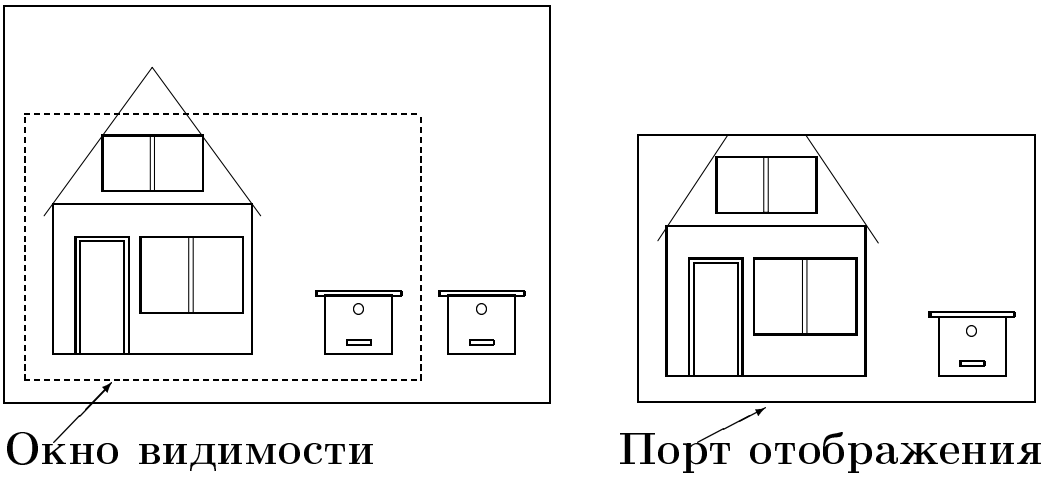
Пусть заданы три точки P_1, P_2, P_3. Найти матрицу преобразования такого, что после преобразования вектор $\vec{P_1P_2}$ будет направлен вдоль оси Z, а вектор $\vec{P_1P_3}$ будет лежать в плоскости YZ.

1. Вначале надо сместить начало координат в точку $\vec{P_1}$ с помощью преобразования
$$T(-x_1, -y_1, -z_1).$$
2. Единичный вектор, который должен лечь вдоль оси Z
$$\vec{R_z} = [r1_z\ r2_z\ r3_z] = \vec{P_1P_2}/|\vec{P_1P_2}|.$$
Здесь $|\vec{P_1P_2}|$ — длина вектора $\vec{P_1P_2}$.
3. Вектор, перпендикулярный плоскости, построенной на векторах $\vec{P_1P_2}$ и $\vec{P_1P_3}$, должен быть направлен вдоль оси X, так как вектор $\vec{P_1P_2}$ лежит вдоль оси Z, а вектор $\vec{P_1P_3}$ лежит в плоскости YZ. Этот вектор задается векторным произведением
$$\vec{R_x} = [r1_x\ r2_x\ r3_x] = \frac{\vec{P_1P_2} \times \vec{P_1P_3}}{|\vec{P_1P_2}| \cdot |\vec{P_1P_3}|}.$$
4. Наконец, вдоль оси Y должен быть направлен вектор, перпендикулярный к векторам $\vec{R_x}$ $\vec{R_z}$:
$$\vec{R_y} = [\ r1_y\ r2_y\ r3_y\ ] = \vec{R_z} \times \vec{R_x}.$$
5. Искомая матрица есть
$$M = T(-x_1, -y_1, -z_1) \cdot \begin{bmatrix} r1_x & r1_y & r1_z & 0 \\ r2_x & r2_y & r2_z & 0 \\ r3_x & r3_y & r3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

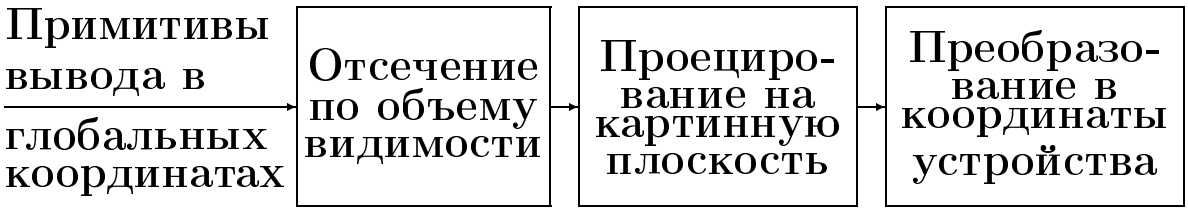
Проецирование

- представление объекта на плоскости отображения (в терминах графики — визуализация объекта)

Визуализация двумерных изображений



Визуализация трехмерных изображений

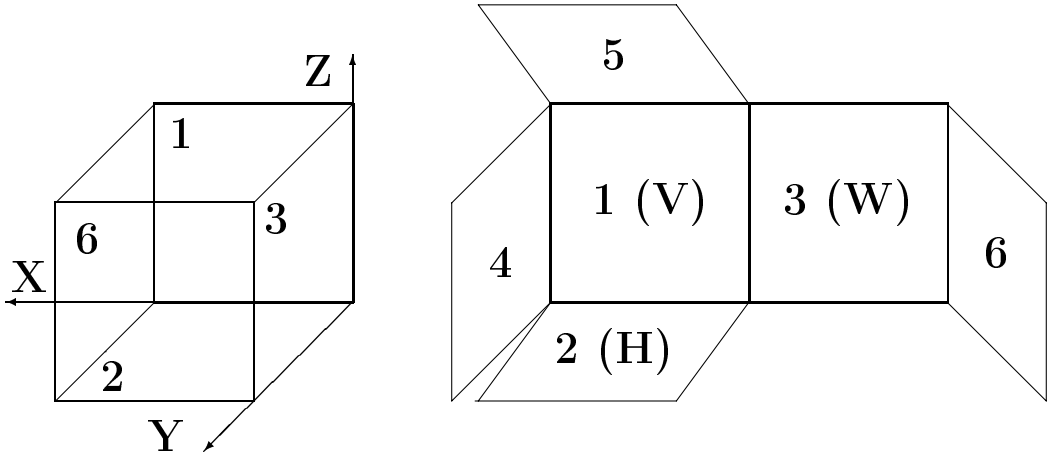


Классификация плоских проекций

- проецирование — отображение точек, заданных в системе координат с размерностью N, в точки в системе меньшей размерности;
- проекторы (проецирующие лучи) — отрезки прямых, идущие из центра проекции через каждую точку объекта до пересечения с плоскостью проекции (картинной плоскостью);

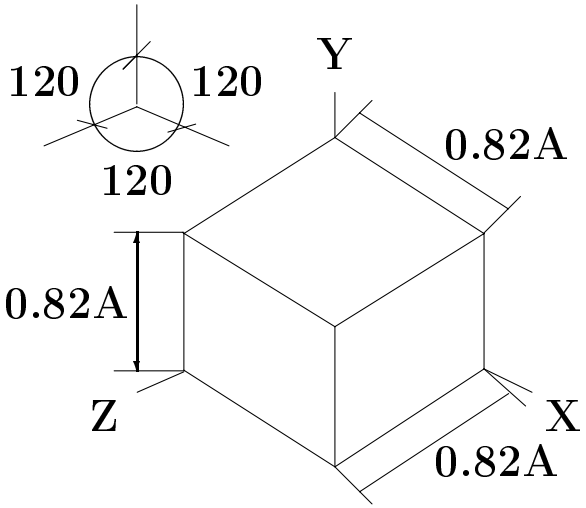
- параллельная проекция — центр проекции на бесконечности от плоскости проекции:
  - ортогональные,
  - прямоугольные аксонометрические — проекторы $\perp$ к плоскости проекции, расположенной под углом к главной оси,
  - косоугольные аксонометрические — плоскость проекции $\perp$ к главной оси, проекторы расположены под углом к плоскости проекции;
- центральная проекция — центр проекции на конечном расстоянии от плоскости проекции. Имеют место т.н. перспективные искажения.

Ортогональные проекции (основные виды)

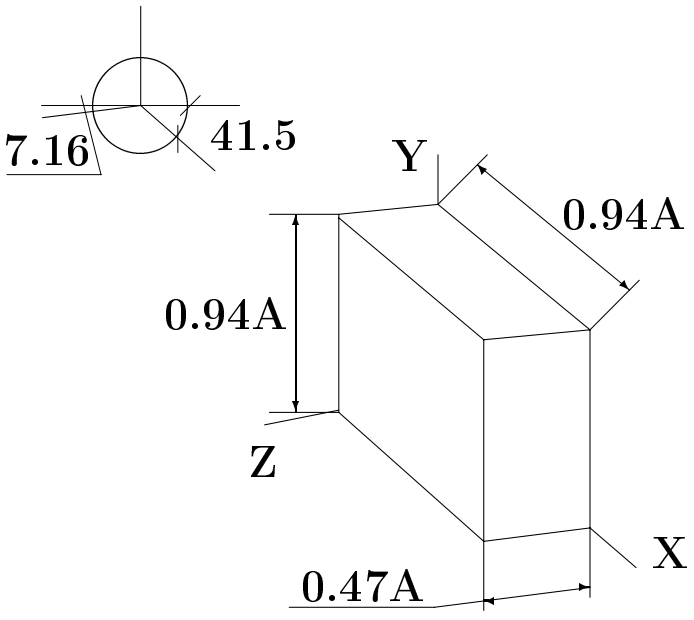


1. Вид спереди, главный вид, фронтальная проекция, (на заднюю грань V),
2. Вид сверху, план, горизонтальная проекция, (на нижнюю грань H),
3. Вид слева, профильная проекция, (на правую грань W),
4. Вид справа (на левую грань),
5. Вид снизу (на верхнюю грань),
6. Вид сзади (на переднюю грань).

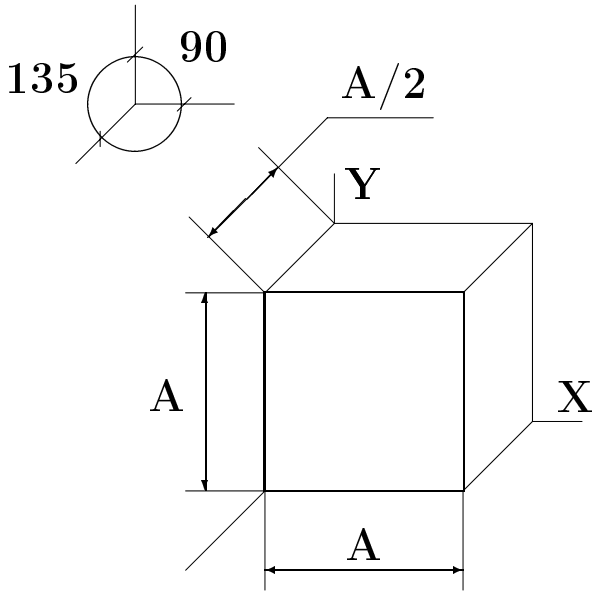
Аксонометрическая прямоугольная изометрическая проекция



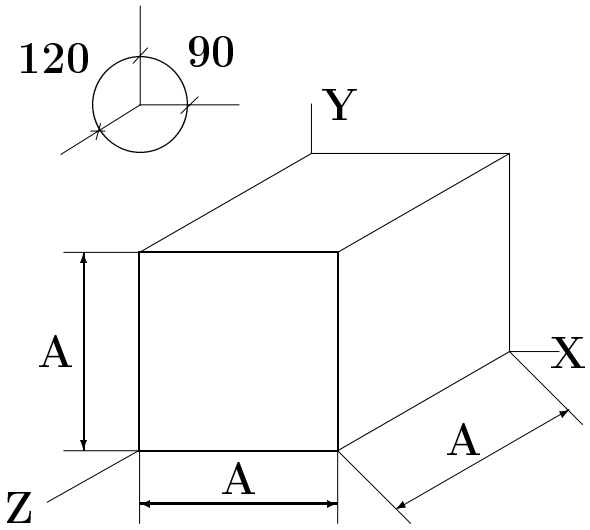
Аксонометрическая прямоугольная диметрическая проекция



Аксонометрическая косоугольная фронтальная диметрическая проекция (проекция кабине)



Аксонометрическая косоугольная горизонтальная изометрическая проекция (военная перспектива)

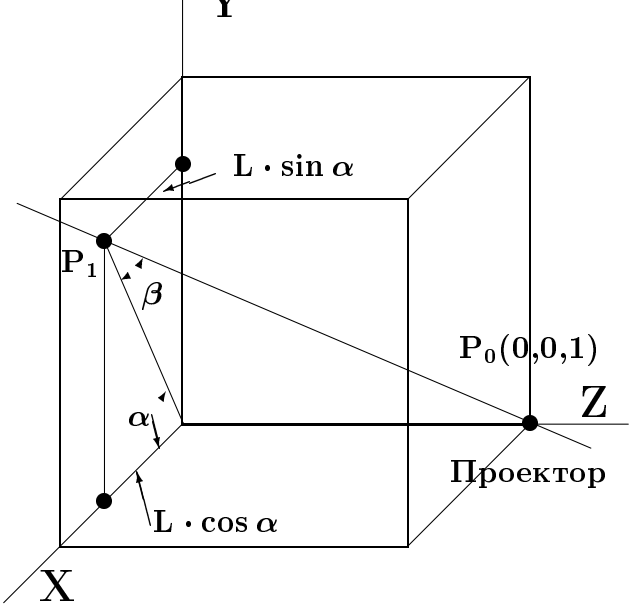


Матрицы преобразования для параллельных проекций

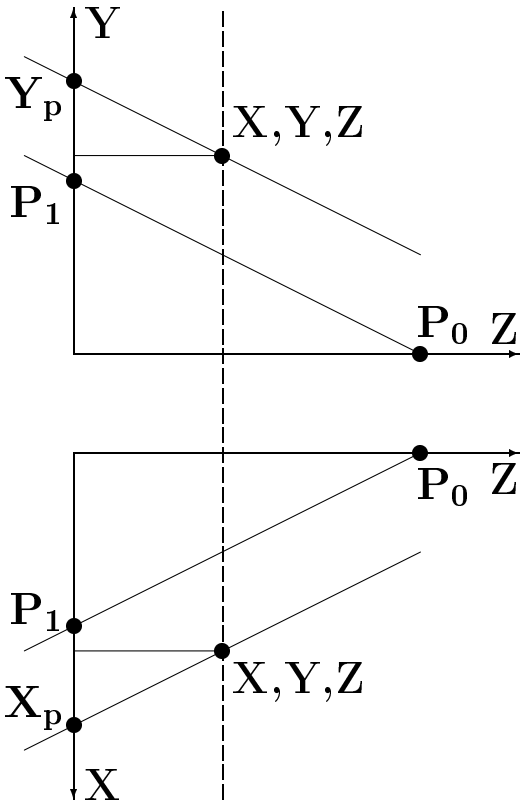
Ортогональное проецирование — на плоскость, перпендикулярную какой-либо оси, т.е. проецирование вдоль этой оси. В частности, проецирование в XY-плоскость, заданную соотношением $Z = Z_0$, обеспечивается матрицей:

$$\begin{bmatrix} xn & yn & zn & wn \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Z_0 & 1 \end{bmatrix}.$$

Косоугольное проецирование — на плоскость, перпендикулярную главной оси, проекторы составляют с плоскостью проецирования угол не равный 90°.



Косоугольная параллельная проекция $P_1(L\cdot\cos\alpha, L\cdot\sin\alpha, 0)$ точки $P_0(0,0,1)$



Проектором, параллельным P_0P_1 спроецируем некоторую точку (X,Y,Z) в точку (X_p, Y_p, Z_p).

Из подобия треугольников получаем:

$(X_p - X)/Z = L\cdot\cos\alpha \Rightarrow$
$X_p = X + Z\cdot L\cdot\cos\alpha$

$(Y_p - Y)/Z = L\cdot\sin\alpha \Rightarrow$
$Y_p = Y + Z\cdot L\cdot\sin\alpha$

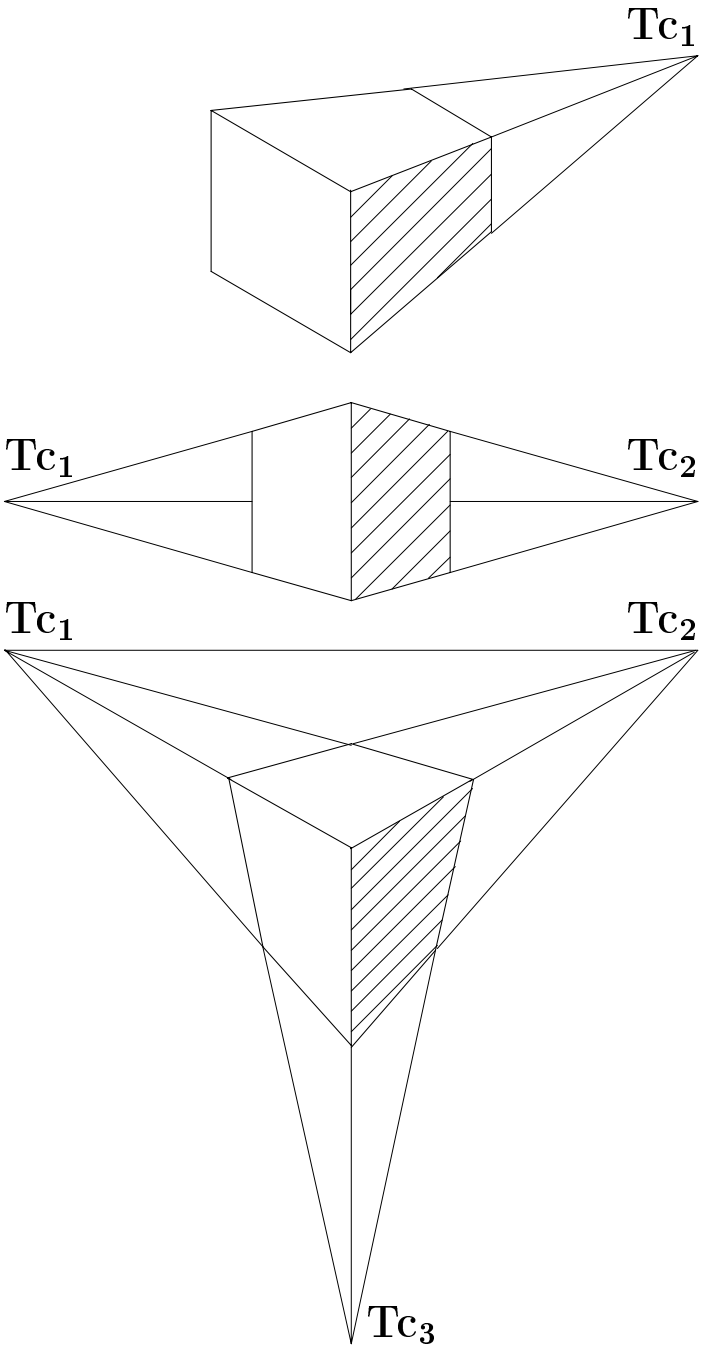
Этому соответствует матричное выражение:

$$\begin{bmatrix} xp & yp & zp & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ L\cdot\cos\alpha & L\cdot\sin\alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

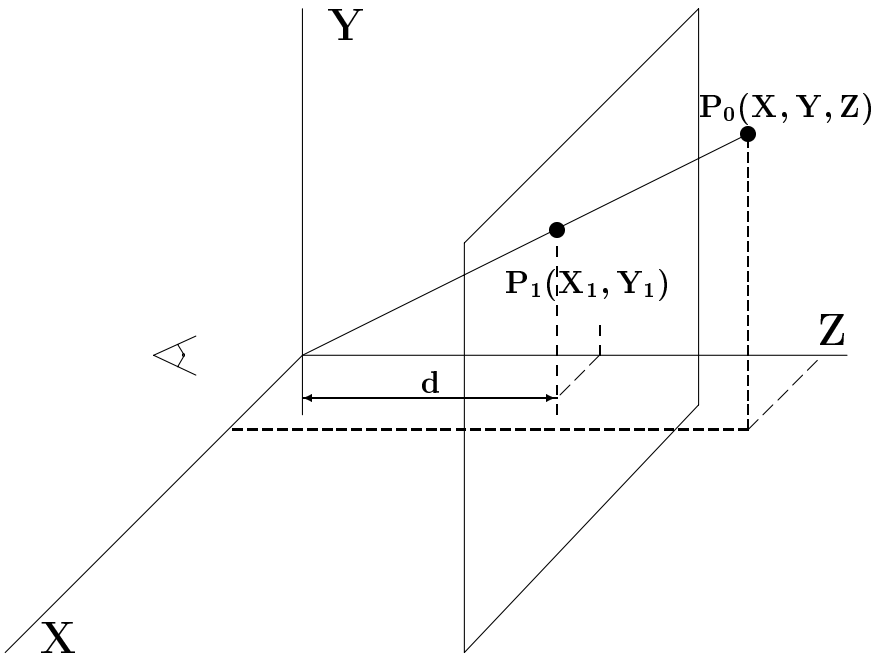
Центральная проекция

Центральные проекции параллельных прямых, не параллельных плоскости проекции сходятся в точке схода.

В зависимости от числа координатных осей, которые пересекает плоскость проекции, различаются одно, двух и трехточечные центральные проекции.



Центральная проекция точки P_0 в плоскость $Z = d$



Соотношения для координат точки P_1, полученной проецированием точки $P_0(X, Y, Z)$ в плоскость $Z = d$:

$$\frac{X_1}{d} = \frac{X}{Z}, \qquad \frac{X_1}{d} = \frac{X}{Z}, \qquad X_1 = \frac{X}{Z/d}, \qquad Y_1 = \frac{Y}{Z/d}.$$

Или, в матричном виде:

$$\begin{bmatrix} x_1 & y_1 & z_1 & w_1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot M = \begin{bmatrix} x & y & z & z/d \end{bmatrix}.$$

Переходя к декартовым координатам, получаем:

$$\begin{bmatrix} \dfrac{X}{(Z/d)} & \dfrac{Y}{(Z/d)} & d & 1 \end{bmatrix}.$$

Если же точка просмотра расположена в плоскости проекции, тогда центр проекции расположен в точке (0, 0, -d).

Аналогично рассмотренному выше, получаем:

$$X_1 = \frac{X}{Z/d + 1}; \qquad Y_1 = \frac{Y}{Z/d + 1}.$$

Матрица преобразования в этом случае имеет вид:

$$M_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица M_0 может быть представлена в виде:

$$M_0 \quad = \quad T(0, 0, d) \cdot M \cdot T(0, 0, -d),$$

т.е. перенос начала координат в центр проецирования, собственно проецирование и обратный сдвиг начала координат.

Стереоизображения

Статические псевдостереоизображения — просмотр фотографий перспективных видов для левого и правого глаз через стереоскоп.

Для разделения динамических видов на экране дисплея могут использоваться:

- цветовое разделение;
- пространственное-временное разделение на дисплеях с чересстрочной разверткой;
- временное разделение на дисплеях с двойным кадровым буфером.

Матрицы преобразований для получения стереопроекций — матрицы перспективных проекций.

- преобразования сдвига,
- преобразования масштабирования,
- преобразования поворота.

Преобразование сдвига — переписывание части изображения (bitblt-операции — Bit Block Transfer). Сдвиг может сопровождаться исполнением операция над кодами пикселов. Наиболее употребимыми являются:

- замена — новый пиксел просто заменяет старый,
- исключающее ИЛИ — в видеопамять заносится результат операции XOR над старым и новым кодами пикселов.

При реализации техники “акварель” (работа с прозрачными цветами), в видеопамять заносится результат цветовой интерполяции между старым и новым оттенками пикселов. Эта операция точно реализуема в полноцветных дисплеях и не обязательно точно на дисплеях с таблицей цветности.

Преобразование масштабирования:

- целочисленное — zoom,
- произвольное, когда коэффициент масштабирования не обязательно целое число, — transfocation.

Нецелочисленное уменьшение

Преобразование поворота

Проблемы возникают при повороте растровой картины на угол не кратный 90°. Здесь возможны два подхода:

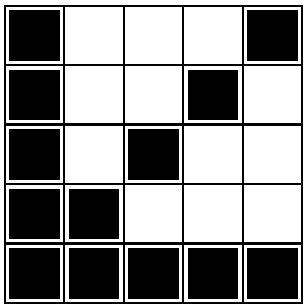
1. **Сканируются строки исходной картины при этом вычисляются новые значения координат пикселов для результирующей картины. Ясно что отсутствие дырок на результирующем изображении может быть обеспечено только при использовании вещественной арифметики, кроме этого возможно повторное занесение пикселов.**
2. **Сканируются строки результирующей картины и по координатам очередного пиксела определяются координаты пиксела из исходного изображения. Этот подход гарантирует отсутствие дырок, кроме того исключает повторное занесение пикселов.**

Генерация отрезков

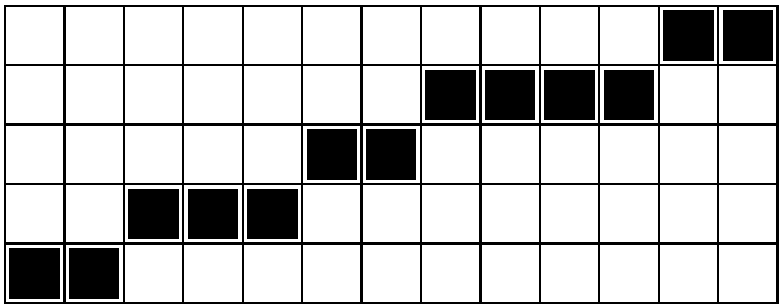
- два алгоритма ЦДА — цифрового дифференциального анализатора (DDA — Digital Differential Analyzer) для генерации векторов — обычный и несимметричный;
- алгоритм Брезенхема для генерации векторов;
- алгоритм Брезенхема для генерации ребер заполненного многоугольника с уменьшением ступенчатости.

Требования к изображению отрезка:

- точное позиционирование концов отрезка;
- отрезки должны выглядеть прямыми;
- яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона;
- должны быть поддержаны требуемые атрибуты.



а)



б)

$$I = I_{pix} / L_{pix}$$

$$I = I_{pix} / (1.41 \times L_{pix})$$

Методы улучшения аппроксимации:

- объективное улучшение аппроксимации — увеличением разрешения дисплея;
- субъективное улучшение аппроксимации — учет психофизиологических особенностей зрения (уменьшение размеров экрана, размывание резких границ).

Цифровой дифференциальный анализатор

Отрезок описывается уравнением:

$$\frac{dY}{dX} = \frac{Py}{Px}$$

ЦДА формирует дискретную аппроксимацию решения этого дифференциального уравнения.

Задается N — количество узлов аппроксимации отрезка и за N циклов вычисляются очередные координаты:

$$X_0 = Xn; \quad X_{i+1} = X_i + Px/N.$$

$$Y_0 = Yn; \quad Y_{i+1} = Y_i + Py/N.$$

X_i, Y_i преобразуются в целочисленные значения.

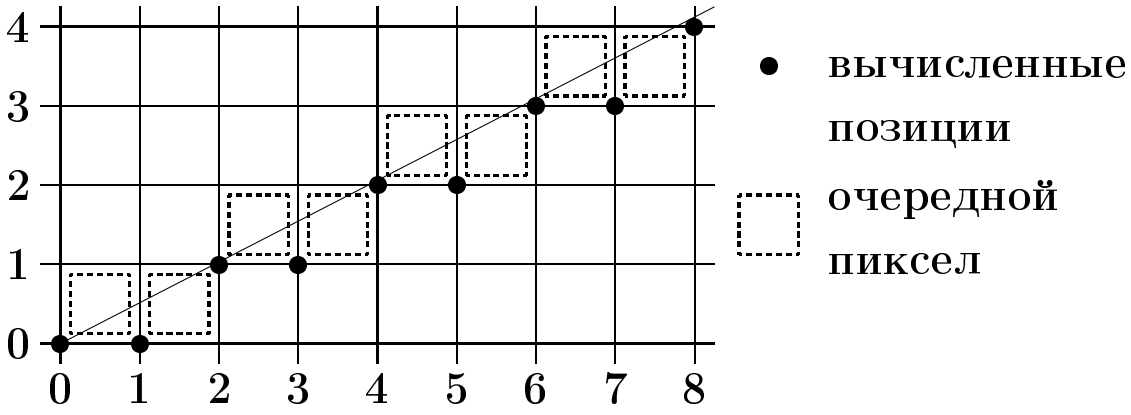
Недостатки:

- точки могут прописываться дважды,
- нет предпочтительных направлений и отрезки кажутся некрасивыми.

Несимметричный ЦДА

По большей координате делается единичный шаг.

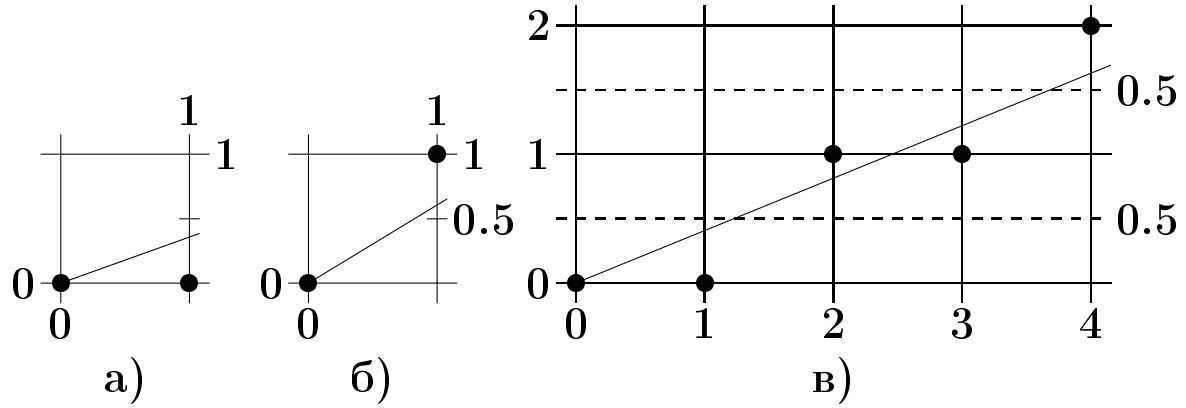
Для Px > Py (Px, Py > 0) X-координата увеличивается на 1 Px раз, при этом Y-координата Px раз увеличивается на Py/Px.



Проблема заключается в необходимости деления (Py/Px) и сложения вещественных чисел.

Алгоритм Брезенхема построения векторов

Алгоритм средней точки. Вещественных операций нет.
Рис. а) и б). Пусть начальная точка (0,0).
Если $P_Y/P_X \le 0.5$, то следующая точка (1,0).
Если $P_Y/P_X > 0.5$, то следующая точка (1,1).



$$X_0 = X_n;\quad Y_0 = Y_n;\quad X_1 = X_0 + 1;\quad E_1 = \frac{\Delta Y}{\Delta X} - \frac{1}{2}.$$

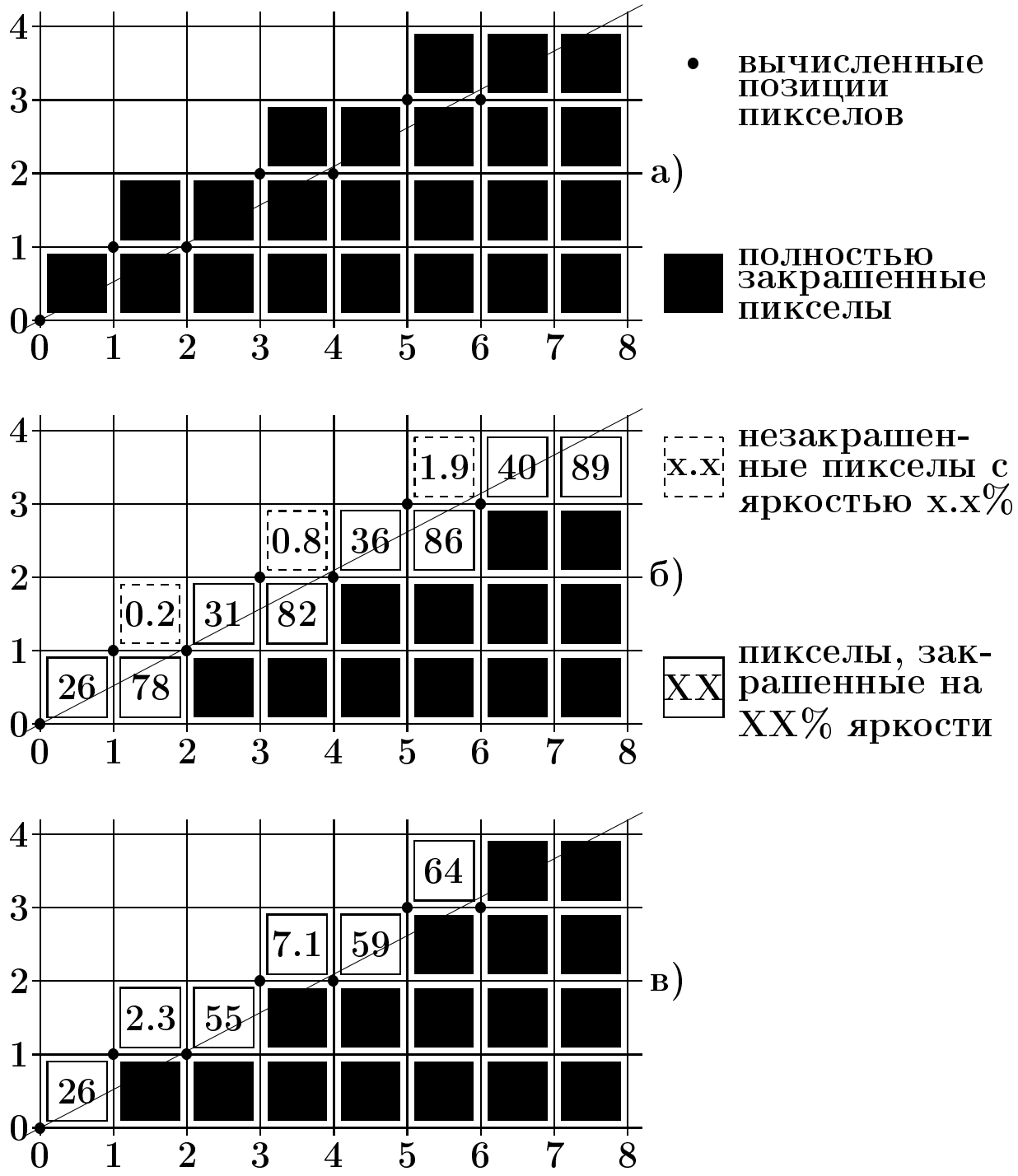
Возможны случаи:

$E_1 \le 0$	$E_1 > 0$
ближайшая точка есть:	
$X_1 = X_0 + 1;$	$X_1 = X_0 + 1;$
$Y_1 = Y_0;$	$Y_1 = Y_0 + 1;$
$E_2 = E_1 + P_Y/P_X;$	$E_2 = E_1 + P_Y/P_X - 1.$

Важен только знак E, поэтому домножим E на $2 \times P_X$:

$$E_1 = 2 \times P_Y - P_X$$
$$E_1 \le 0:\ E_2 = E_1 + 2 \times P_Y$$
$$E_1 > 0:\ E_2 = E_1 + 2 \times (P_Y - P_X)$$

Модифицированный алгоритм Брезенхема



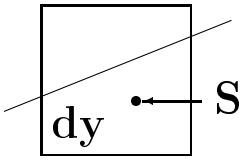
а) генерация ребер без устранения ступенчатости;

б) точное вычисление интенсивности пикселов;

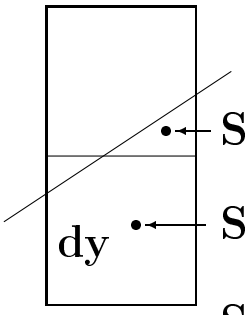
в) пикселы границы по модифицированному методу Брезенхема.

Построение ребра заполненного многоугольника с устранением ступенчатости.

Яркость пиксела $\sim$ площади пиксела, попавшей внутрь многоугольника.



а)



б)

dy — отсекаемая часть пиксела по Y

t — тангенс угла наклона отрезка

Исходный алгоритм	Модифицированный
$E = t - 1/2$	$E' = E + (1 - t)$
$-1/2 \le E \le 1/2$	$0 \le E' \le 1$

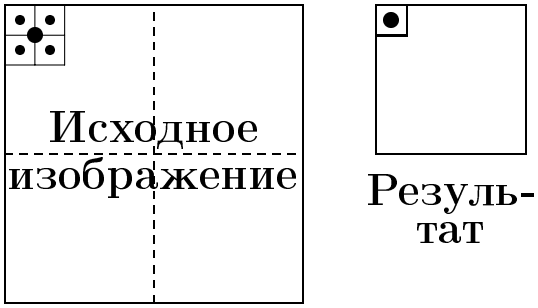
E' — доля площади пиксела внутри многоугольника.

Алгоритм для случая $0 \le dY \le dX$:

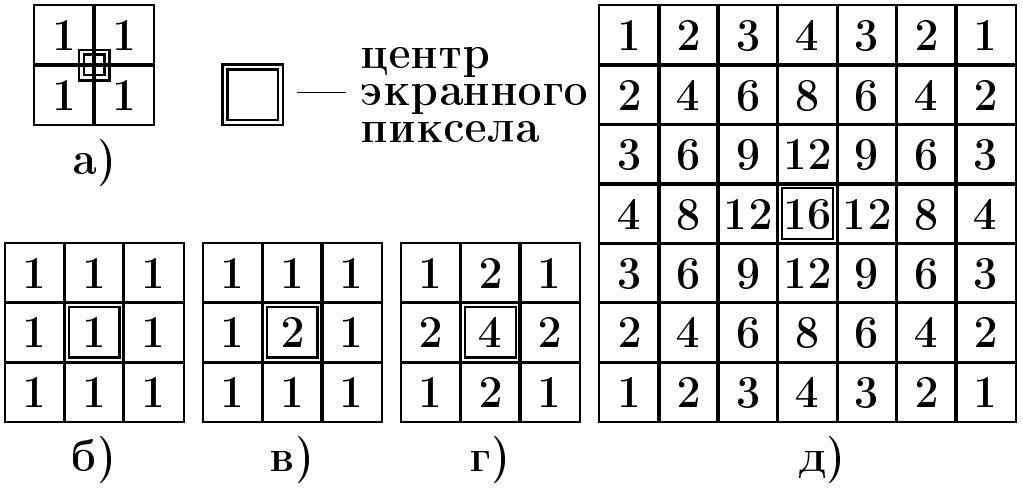
```
X= x1;                          Y= y1;
Px= x2 − x1;                    Py= y2 − y1;
t= I ∗ Py/Px;                   E′= I/2;
E′max= I − I ∗ Py/Px;           i= Px;
PutPixel(X, Y, t/2);        /* Первая точка вектора */
while (i = i − 1 ≥ 0) {
    if (E′ ≥ E′max) {
            X= X + 1;   Y= Y + 1;   E′ = E′ − E′max;
    } else  X= X + 1;
    E′ = E′ + t;
    PutPixel(X, Y, E′);     /* Очередная точка */
}
```

- увеличение разрешения устройства вывода;
- уменьшение размеров устройства вывода;
- усреднение с понижением разрешения:
  $\mathbf{I_{ij} = (\sum_{k=1}^{m} \sum_{l=1}^{m} M_{kl} \times V_{k+(i-1)\times m, l+(j-1)\times m}) / S,}$
  $\mathbf{i = 1, 2, 3 \ldots;\ \ j = 1, 2, 3 \ldots}$
- "размывание" границ без понижения разрешения:
  $\mathbf{I_{ij} = (\sum_{k=1}^{m} \sum_{l=1}^{m} M_{kl} \times V_{k+i-i_0-1, l+j-j_0-1}) / S,}$
  $\mathbf{m\ ,\ \ i_0 = j_0 = (m-1)/2;}$
  $\mathbf{i = i_0 + 1, i_0 + 2, \ldots;\quad j = j_0 + 1, j_0 + 2, \ldots}$

$\mathbf{S = \sum_{k=1}^{m} \sum_{l=1}^{m} M_{kl}}$



Усреднение с понижением размерности в 2 раза



Маски равномерного (а–б)
и взвешенного (в–д) усреднения изображений

Подчеркивание границ

Для подчеркивания границ используется высокочастотная фильтрация исходного изображения с помощью масок вида:

0	-1	0
-1	5	-1
0	-1	0

а)

-1	-1	-1
-1	9	-1
-1	-1	-1

б)

1	-2	1
-2	5	-2
1	-2	1

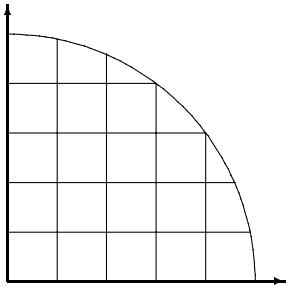
в)

Вычислительная процедура аналогична рассмотренной выше.

Генерация окружности

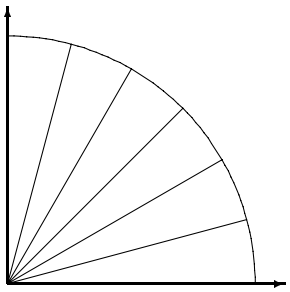
Координатное представление

$$X^2 + Y^2 = R^2$$
$$Y = \sqrt{R^2 - X^2}$$



Параметрическое представление

$$X = R \times \cos\phi$$
$$Y = R \times \sin\phi$$

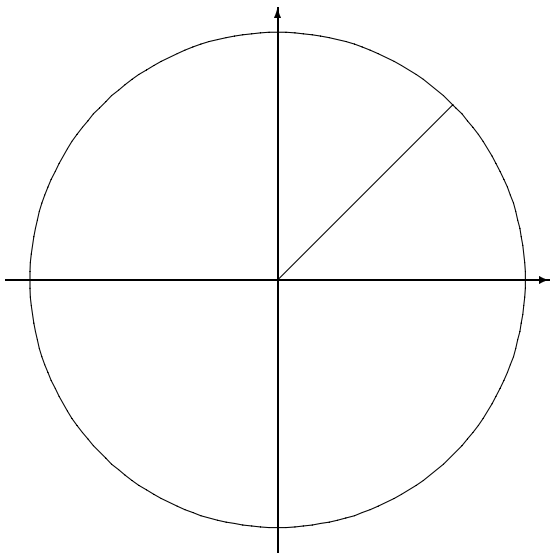


Алгоритм Брезенхема

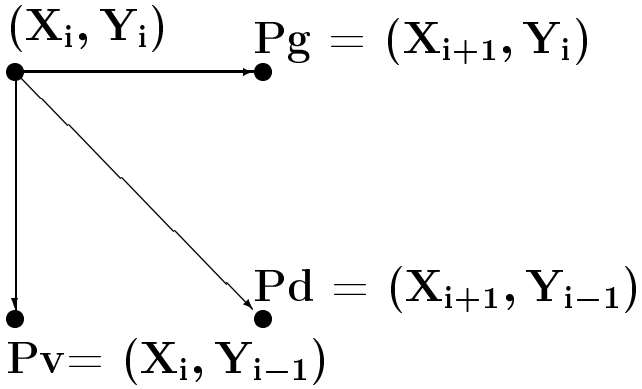
$$X^2 + Y^2 = R^2$$
$$\min E_i(P_i) = (X_i^2 + Y_i^2) - R^2$$

$P_i(X_i, Y_i)$ — очередной пиксел

Достаточно сгенерировать точки для 1/8 окружности



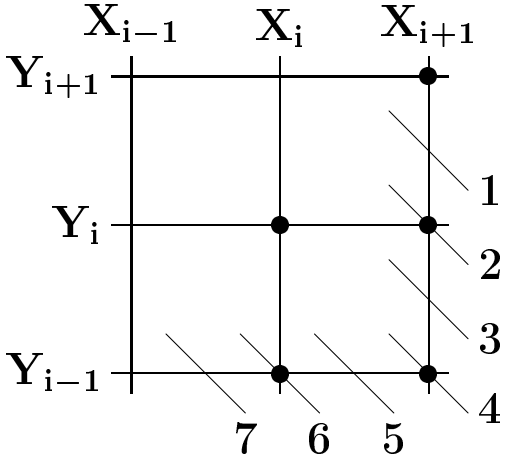
При генерации окружности по часовой стрелке после занесения пиксела (X_i, Y_i) следующий пиксел может быть P_g, P_d или P_v



Для выбора возможного пиксела вычислим:

$$\begin{aligned} |Dg| &= |(X+1)^2 + Y^2 - R^2| \\ |Dd| &= |(X+1)^2 + (Y-1)^2 - R^2| \\ |Dv| &= |X^2 + (Y-1)^2 - R^2| \end{aligned}$$

В качестве очередного пиксела выберем тот, для которого $|D|$ минимально



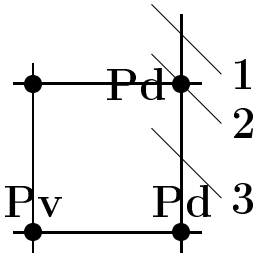
$$\begin{aligned} Dd < 0 &- (1-3) \\ Dd > 0 &- (5-7) \\ Dd = 0 &- (4) \end{aligned}$$

Случай Dd < 0

Выбор между возможными следующими пикселами Pg или Pd определяется разностью:

$$\begin{array}{rl|l} di = & |Dg| - |Dd| = & \leq 0 - Pg \\ & |(X+1)^2 + Y^2 - R^2| - & > 0 - Pd \\ & |(X+1)^2 + (Y-1)^2 - R^2|. & \end{array}$$

Вычисление di. Варианты 2 и 3:



Так как Pg или вне или на окружности, а Pd внутри, то

$Dg \geq 0; \quad Dd < 0.$

Таким образом:

$$di = (X+1)^2 + Y^2 - R^2 + (X+1)^2 + (Y-1)^2 - R^2.$$
$$di = 2 \cdot [(X+1)^2 + (Y-1)^2 - R^2] + 2 \cdot Y - 1.$$
$$di = 2 \cdot (Dd + Y) - 1$$

Вариант 1

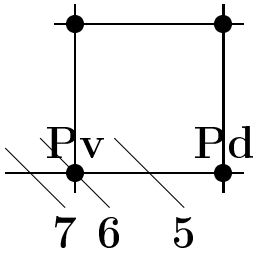
Очевидно должен быть выбран пиксел Pg.
$Dg < 0$, $Dd < 0$ и $|Dg| < |Dd| \Longrightarrow di < 0$, т.е. по критерию $di < 0$ как и ранее д.б. выбран Pg, что верно.

Случай Dd > 0

Выбор между возможными следующими пикселами Pd или Pv определяется разностью:

$$\begin{array}{rl|rl} \mathbf{si} = & |\mathbf{Dd}| - |\mathbf{Dv}| = & \leq & \mathbf{0} - \mathbf{Pd} \\ & |(\mathbf{X}+1)^2 + (\mathbf{Y}-1)^2 - \mathbf{R}^2| - & > & \mathbf{0} - \mathbf{Pv} \\ & |\mathbf{X}^2 + (\mathbf{Y}-1)^2 - \mathbf{R}^2| & & \end{array}$$

Вычисление si. Варианты 5 и 6:



Так как Pd вне, а Pv или внутри или на окружности, то

$\mathbf{Dd} > \mathbf{0}; \quad \mathbf{Dv} \leq \mathbf{0}.$

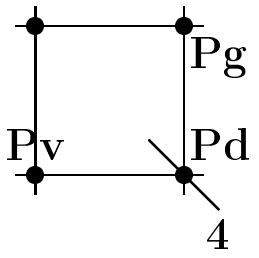
Таким образом:

$$\mathbf{si} = (\mathbf{X}+1)^2 + (\mathbf{Y}-1)^2 - \mathbf{R}^2 + \mathbf{X}^2 + (\mathbf{Y}-1)^2 - \mathbf{R}^2;$$
$$\mathbf{si} = \mathbf{2} \cdot [(\mathbf{X}+1)^2 + (\mathbf{Y}-1)^2 - \mathbf{R}^2] - \mathbf{2} \cdot \mathbf{X} - \mathbf{1};$$
$$\mathbf{si} = \mathbf{2} \cdot (\mathbf{Dd} - \mathbf{X}) - \mathbf{1}.$$

Вариант 7

Очевидно должен быть выбран пиксел Pv.
$\mathbf{Dd} > \mathbf{0}$, $\mathbf{Dv} > \mathbf{0}$ и $|\mathbf{Dd}| > |\mathbf{Dv}| \Longrightarrow \mathbf{si} > \mathbf{0}$, т.е. по критерию $\mathbf{si} > \mathbf{0}$ как и ранее д.б. выбран Pv, что верно.

Случай $Dd = 0$



di: $Dg > 0;\ \ Dd = 0;\ \Longrightarrow$
по критерию $di > 0$ выбираем Pv

si: $Dd = 0;\ \ Dv < 0;\ \Longrightarrow$
по критерию $si \le 0$ выбираем Pv

Итак:

$Dd < 0$: $di \le 0$ — выбор горизонтального пиксела Pg
$di > 0$ — выбор диагонального пиксела Pd

$Dd > 0$: $si \le 0$ — выбор диагонального пиксела Pd
$si > 0$ — выбор вертикального пиксела Pv

$Dd = 0$: выбор диагонального пиксела Pd.

Рекуррентные соотношения для вычисления Dd_{i+1}

1. Для горизонтального шага к X_{i+1}, Y_i
   $X_{i+1} = X_i + 1;\ \ Y_{i+1} = Y_i;$
   $$Dd_{i+1} = (X_{i+1} + 1)^2 + (Y_{i+1} - 1)^2 - R^2 = (X_i + 1)^2 + (Y_i - 1)^2 - R^2 + 2 \cdot X_{i+1} + 1 = Dd_i + 2 \cdot X_{i+1} + 1;$$

2. Для диагонального шага к X_{i+1}, Y_{i-1}
   $X_{i+1} = X_i + 1;\ \ Y_{i+1} = Y_i - 1;$
   $$Dd_{i+1} = Dd_i + 2 \cdot X_{i+1} - 2 \cdot Y_{i+1} + 2;$$

3. Для вертикального шага к X_i, Y_{i-1}
   $X_{i+1} = X_i;\ \ Y_{i+1} = Y_i - 1;$
   $$Dd_{i+1} = Dd_i - 2 \cdot Y_{i+1} + 1;$$

4. Начальная инициализация
   $X = 0;\ \ Y = R;$
   $$Dd = (X+1)^2 + (Y-1)^2 - R^2 = 1 + (R-1)^2 - R^2 = 2 * (1 - R)$$

Линии и поверхности

- аналитически заданные (кривые) линии,
- интерполяционные кривые.

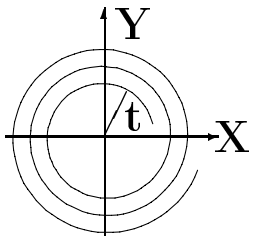
Аналитически заданные линии

- компактное представление,
- легкое вычисление различных параметров (наклон и т.п.),
- легкое отображение,
- легкое вычисление всех точек линии,
- легкая модификация.

Представление аналитически заданных линий

- явное или неявное непараметрическое (зависит от выбора координат),
- параметрическое.

: $Z = F(X, Y)$.
: $\Phi(X, Y, Z) = 0$.
: $X = \phi(t);\ Y = \psi(t);\ Z = \omega(t)$.



- возможно задание неоднозначных кривых,
- нет проблем с заданием производных.

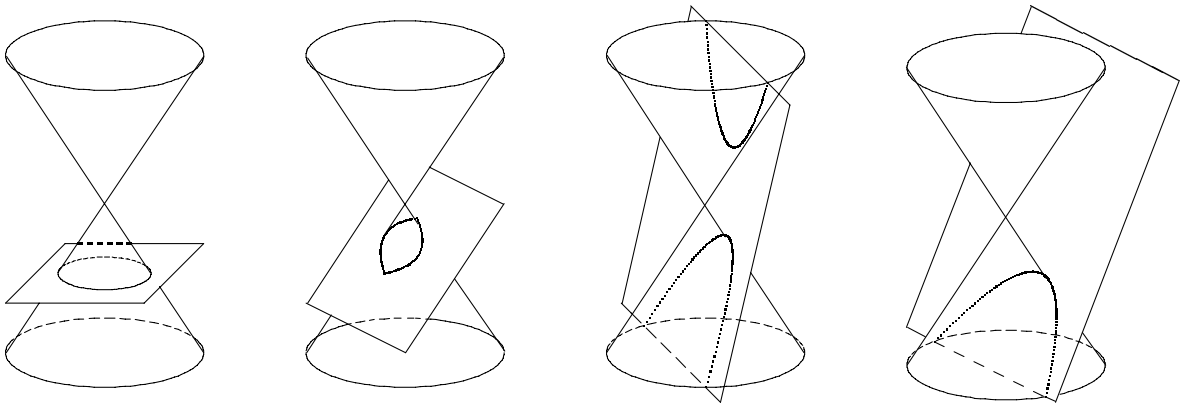
Интерполяционные (кривые) линии

- известны значения функции в опорных точках,
- отыскиваются значения в промежуточных точках.

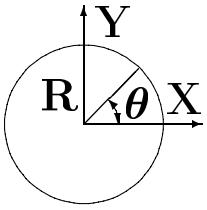
Конические сечения

Описываются уравнением 2-й степени:

$$a \cdot X^2 + b \cdot X \cdot Y + c \cdot Y^2 + d \cdot X + e \cdot Y + f = 0$$

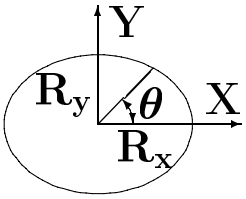


$$\left(\frac{X}{R}\right)^2 + \left(\frac{Y}{R}\right)^2 = 1;$$



$$X = R\cos\theta; \quad Y = R\sin\theta; \quad -\pi \le \theta \le \pi.$$

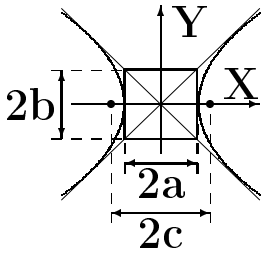
$$\left(\frac{X}{R_x}\right)^2 + \left(\frac{Y}{R_y}\right)^2 = 1;$$



$$X = R_x\cos\theta; \quad Y = R_y\sin\theta; \quad -\pi \le \theta \le \pi.$$

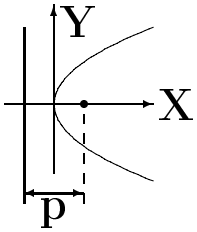
$$\left(\frac{X}{a}\right)^2 - \left(\frac{Y}{b}\right)^2 = 1; \quad b^2 = c^2 - a^2;$$

$$y = \frac{b}{a}x \; - \quad ;$$



$$X = \pm a\sec\theta; \quad Y = \pm b\tan\theta; \quad 0 \le \theta \le \pi/2.$$

$$Y^2 = 2pX;$$



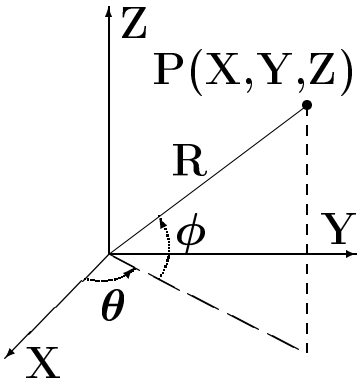
$$X = 2p\theta^2; \quad 0 \le \theta \le \infty;$$

$$Y = \pm 2p\theta.$$

Квадратичные поверхности

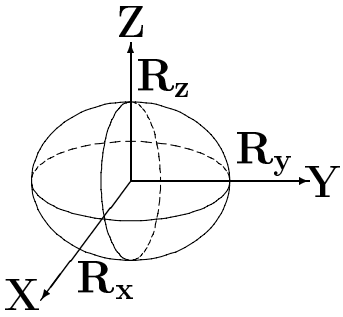
$$\left(\frac{X}{R}\right)^2 + \left(\frac{Y}{R}\right)^2 + \left(\frac{Z}{R}\right)^2 = 1;$$

$$X = R\cos\phi\cos\theta; \quad -\pi \le \theta \le \pi;$$
$$Y = R\cos\phi\sin\theta; \quad -\pi/2 \le \phi \le \pi/2;$$
$$Z = R\sin\phi$$



$$\left(\frac{X}{R_x}\right)^2 + \left(\frac{Y}{R_y}\right)^2 + \left(\frac{Z}{R_z}\right)^2 = 1;$$

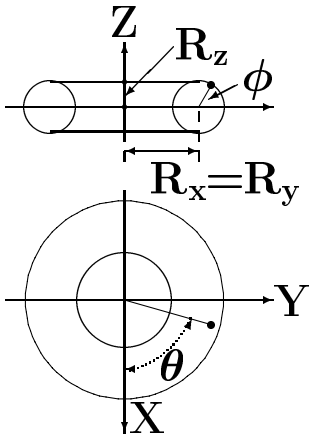
$$X = R_x\cos\phi\cos\theta; \quad -\pi \le \theta \le \pi;$$
$$Y = R_y\cos\phi\sin\theta; \quad -\pi/2 \le \phi \le \pi/2;$$
$$Z = R_z\sin\phi$$



$$\left[\, R - \sqrt{\left(\frac{X}{R_x}\right)^2 + \left(\frac{Y}{R_y}\right)^2}\,\right]^2 + \left(\frac{Z}{R_z}\right)^2 = 1;$$

$$X = R_x(R + \cos\phi)\cos\theta; \quad -\pi \le \theta \le \pi;$$
$$Y = R_y(R + \cos\phi)\sin\theta; \quad -\pi \le \phi \le \pi;$$
$$Z = R_z\sin\phi;$$

R –



Суперквадрики

Обобщение квадратичных кривых введением дополнительных параметров, управляющих внешним видом кривой. Число дополнительных параметров равно размерности (1 — для кривой, 2 — для поверхности).

Суперэллипс

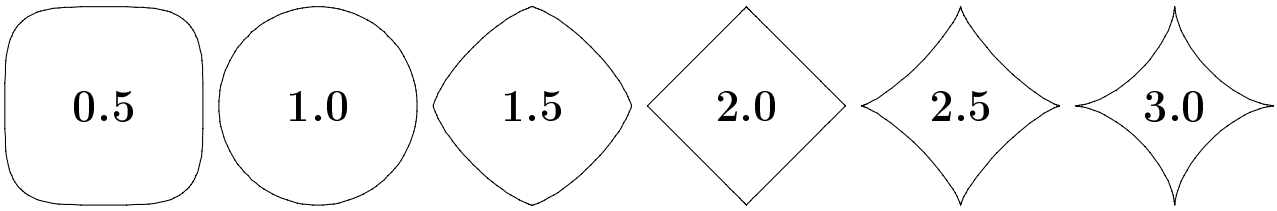
В декартовых координатах:

$$\left(\frac{X}{R_x}\right)^{2/s} + \left(\frac{Y}{R_y}\right)^{2/s} = 1$$

В параметрическом представлении:

$$X = R_x \cdot \cos^s \theta; \quad Y = R_y \cdot \sin^s \theta; \quad -\pi \le \theta \le \pi.$$

Суперэллипс для различных значений s:



Суперэллипсоид

В декартовых координатах:

$$\left[\left(\frac{X}{R_x}\right)^{2/s_2} + \left(\frac{Y}{R_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{Z}{R_z}\right)^{2/s_1} = 1$$

В параметрическом представлении:

$$X = R_x \cdot \cos^{s_1} \phi \cdot \cos^{s_2} \theta; \qquad -\pi \le \theta \le \pi;$$
$$Y = R_y \cdot \cos^{s_1} \phi \cdot \sin^{s_2} \theta; \qquad -\pi/2 \le \phi \le \pi/2;$$
$$Z = R_z \cdot \sin^{s_1} \phi.$$

Интерполяционные кривые

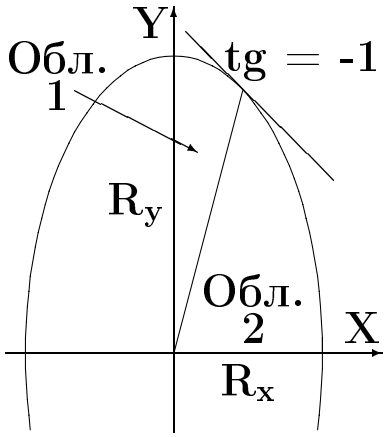
Наиболее широко используются параметрические кубические кривые, имеющие вид:

$$\begin{aligned} x(t) &= A_{11}\,t^3 + A_{12}\,t^2 + A_{13}\,t + A_{14}; \\ y(t) &= A_{21}\,t^3 + A_{22}\,t^2 + A_{23}\,t + A_{24}; \quad 0 \le t \le 1 \\ z(t) &= A_{31}\,t^3 + A_{32}\,t^2 + A_{33}\,t + A_{34}. \end{aligned}$$

Причины популярности:

- можно обеспечить непрерывность функции, значения первой (второй) производной в точках сшивки сегментов кривых,
- можно задать неплоскую кривую.

Генерация эллипса



$$\left(\frac{x}{R_x}\right)^2 + \left(\frac{y}{R_y}\right)^2 = 1.$$

$$E(x,y) = R_y^2x^2 + R_x^2y^2 - R_x^2R_y^2 = 0.$$

$$E(x,y)\begin{cases} < 0 & (x,y) \\ = 0 & (x,y) \\ > 0 & (x,y) \end{cases}$$

Область 1 — шаг по x=1. Область 2 — шаг по y=1

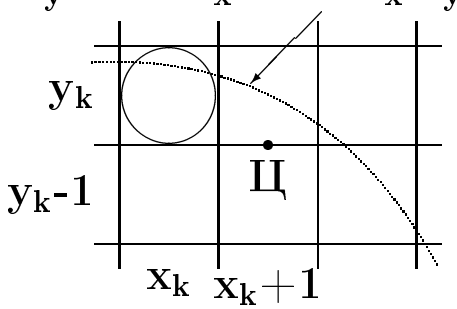
Условие перехода из области 1 в область 2 — значение производной ≤ -1:

$$dy/dx = -2R_y^2x/2R_x^2y = -1 \Longrightarrow$$

$$R_y^2x \ge R_x^2y, \quad 1\ \ 2$$

Эллипс в области 1

$$R_y^2x^2 + R_x^2y^2 - R_x^2R_y^2 = 0$$



$$\begin{aligned} E1_k &= E(x_k+1, y_k-1/2) \\ &= R_y^2(x_k+1)^2 + \\ &\quad R_x^2(y_k-1/2)^2 - \\ &\quad R_x^2R_y^2 \end{aligned}$$

$$E1_k < 0: \quad y_{k+1} = y_k$$
$$E1_k \ge 0: \quad y_{k+1} = y_k - 1$$

Отклонение на шаге от x_{k+1} к $x_{k+2} = x_k + 2$ равно:

$$
\begin{aligned}
E1_{k+1} &= E(x_{k+1} + 1,\ y_{k+1} - 1/2) \\
&= R_y^2\left((x_k + 1) + 1\right)^2 + R_x^2\left(y_{k+1} - 1/2\right)^2 - R_x^2 R_y^2 \\
&= \underline{R_y^2(x_k + 1)^2} + 2R_y^2(x_k + 1) + R_y^2 + \\
&\quad \underline{R_x^2\left(y_k - 1/2\right)^2} - R_x^2\left(y_k - 1/2\right)^2 + \\
&\quad R_x^2\left(y_{k+1} - 1/2\right)^2 - \underline{R_x^2 R_y^2} \\
&= E1_k + 2R_y^2(x_k + 1) + R_y^2 + \\
&\quad R_x^2\left[\left(y_{k+1} - 1/2\right)^2 - \left(y_k - 1/2\right)^2\right]
\end{aligned}
$$

–	y_{k+1}		–
$E1_k < 0$:	y_k	$2R_y^2 x_{k+1} + R_y^2$	
$E1_k \geq 0$:	$y_k - 1$	$2R_y^2 x_{k+1} + R_y^2 - R_x^2 y_{k+1}$	

Отклонение вычисляется с использованием только операций сложения и вычитания, так как компоненты $2R_y^2 x$ и $2R_x^2 y$ определяются приращениями.

Компонент	Стартовое значение в точке $(0, R_y)$	Приращение
$2xR_y^2$	0	$2R_y^2$
$2yR_x^2$	$2R_x^2 R_y$	$2R_x^2$

При каждом приращении проверяется неравенство, задающее переход в область 2:

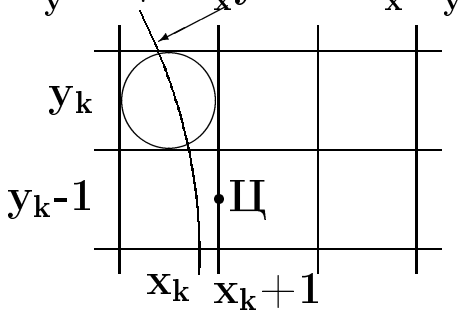
$$R_y^2 x \geq R_x^2 y$$

Начальное значение отклонения:

$$
\begin{aligned}
E1_0 &= E\left(1, R_y - \tfrac{1}{2}\right) \\
&= R_y^2 + R_x^2\left(R_y - \tfrac{1}{2}\right)^2 - R_x^2 R_y^2 = R_y^2 - R_x^2 R_y + \tfrac{1}{4}R_x^2
\end{aligned}
$$

Эллипс в области 2

$R_y^2x^2 + R_x^2y^2 - R_x^2R_y^2 = 0$



$$\begin{aligned} E2_k &= E(x_k + 1/2, y_k - 1) \\ &= R_y^2(x_k + 1/2)^2 + \\ &\quad R_x^2(y_k - 1)^2 - \\ &\quad R_x^2R_y^2 \end{aligned}$$

$$E2_k > 0: \quad x_{k+1} = x_k$$
$$E2_k \le 0: \quad x_{k+1} = x_k + 1$$

Отклонение на шаге от y_{k+1} к $y_{k+1} - 1 = y_k - 2$:

$$\begin{aligned} E2_{k+1} &= E(x_{k+1} + 1/2,\ y_{k+1} - 1) \\ &= R_y^2\left(x_{k+1} + 1/2\right)^2 + R_x^2\left[(y_k - 1) - 1\right]^2 - R_x^2R_y^2 \\ &= E2_k + 2R_x^2(y_k - 1) + R_x^2 + \\ &\quad R_y^2\left[(x_{k+1} + 1/2)^2 - (x_k + 1/2)^2\right] \end{aligned}$$

здесь x_{k+1} равно x_k или $x_k + 1$ в зависимости от знака $E2_k$

Последняя точка области 1 является начальной точкой x_0, y_0 в области 2, поэтому начальное значение отклонения для области 2 равно:

$$\begin{aligned} E2_0 &= E\left(x_0 + 1/2, y_0 - 1\right) \\ &= R_y^2\left(x_0 + 1/2\right)^2 + R_x^2\left(y_0 - 1\right)^2 - R_x^2R_y^2 \end{aligned}$$

$$E2_{k+1} = \begin{cases} E2_k - 2R_x^2y_{k+1} - R_x^2 & E2_k > 0 \\ E2_k - 2R_x^2y_{k+1} - R_x^2 + 2R_y^2x_{k+1} & E2_k \le 0 \end{cases}$$

Закраска областей

Разновидности заполнения (для растровых устройств):

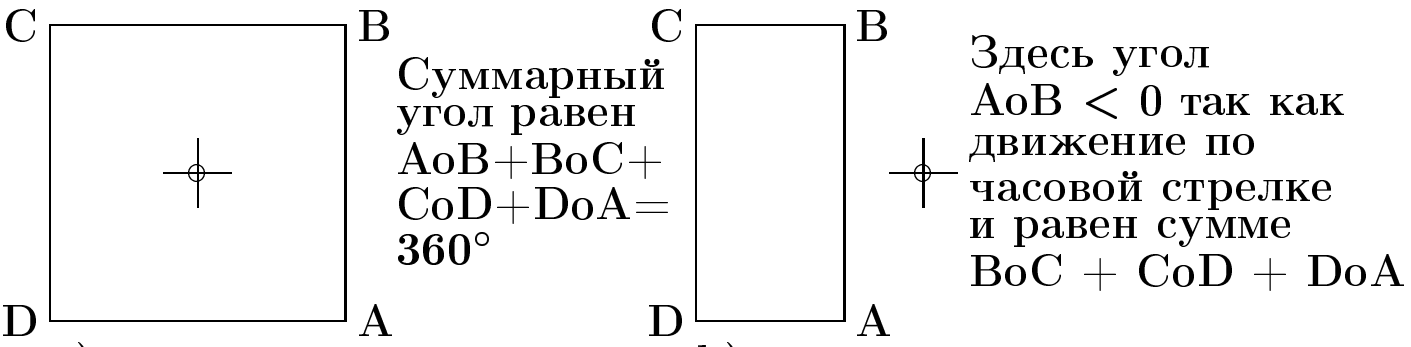
- заполнение внутренней части многоугольника, заданного координатами его вершин.
- заливка области, которая либо очерчена границей с кодом пиксела, отличающимся от кодов любых пикселов внутри области, либо закрашена пикселами с заданным кодом;

Заполнение многоугольника

Способы:

1. Определением принадлежности пиксела экрана многоугольнику.
2. Штриховкой.
3. Ускоренный метод построчного заполнения.

Определение принадлежности пиксела многоугольнику



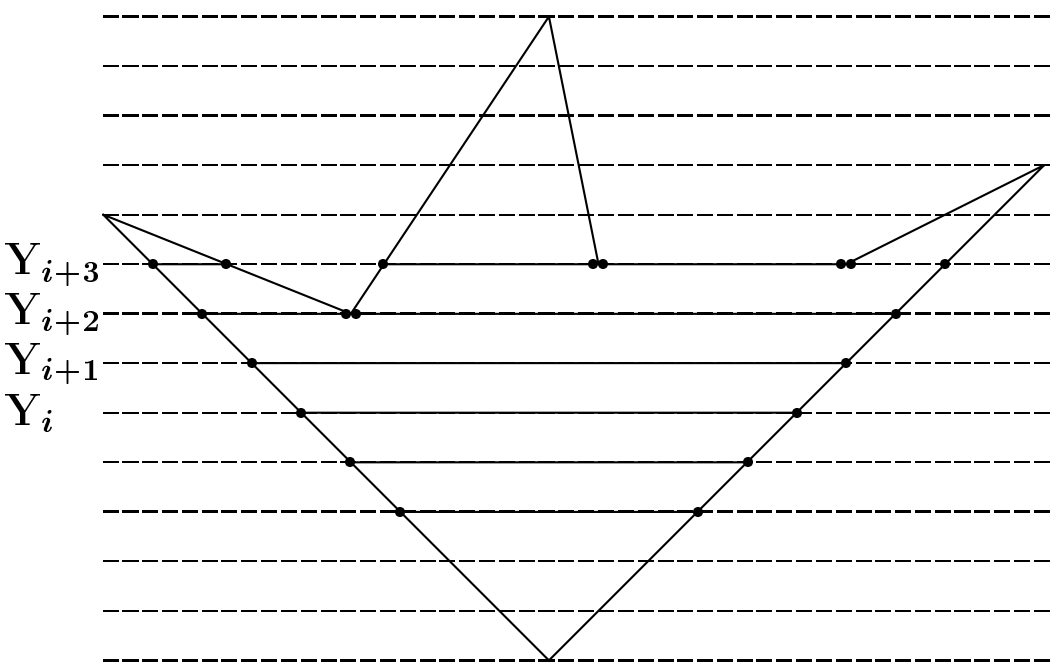
a) точка o внутри

b) точка o вне

a) Пиксел внутри — суммарный угол = 360°

b) Пиксел снаружи — суммарный угол = 0°

Штриховка



Для каждой строки сканирования:

1. Вычисляются X-координаты пересечений со всеми ребрами.
2. X-координаты пересечений сортируются.
3. Закраска ведется между парами отсортированных координат.

Ускоренное построчное заполнение

1. Смена интервалов закраски происходит только тогда, когда в строке сканирования появляется вершина.
2. Зная X-координату пересечения с i-ой строкой легко вычислить пересечение с $i+1$-ой:

$$X_{i+1} = X_i + 1/k$$

где $k = dy/dx$ — тангенс угла наклона ребра

Для каждой строки рассматриваются только те ребра, которые пересекают строку. Они задаются списком активных ребер (САР). При переходе к следующей строке перевычисляются X-координаты пересечений. При появлении в строке сканирования вершин производится перестройка САР.

Общая схема алгоритма:

1. Подготовить массивы Y-координат вершин и номеров вершин.
2. Совместно отсортировать Y-координаты по возрастанию и массив номеров вершин.
3. Определить пределы заполнения по оси Y — Y_min и Y_max. Стартуя с текущим значением Y_tek=Y_min, исполнять пункты 4–9 до завершения раскраски.
4. Определить число вершин, расположенных на текущей строке Y_tek.
5. Если вершины есть, то для каждой из вершин дополнить список активных ребер.
   Для каждого ребра в список активных ребер заносятся:
   - Y_{max} ребра,
   - dX при увеличении Y на 1,
   - $X_{\text{начальное}}$ ребра.

   Горизонтальные ребра тут же закрашиваются и информация о них в список активных ребер не заносится.
   Если после этого обнаруживается, что список активных ребер пуст, то заполнение закончено.

6. По списку активных ребер определяется Y_след — Y-координата ближайшей вершины. (До Y_след нет модификации САР а только меняются X-координаты пересечений строки сканирования с активными ребрами).

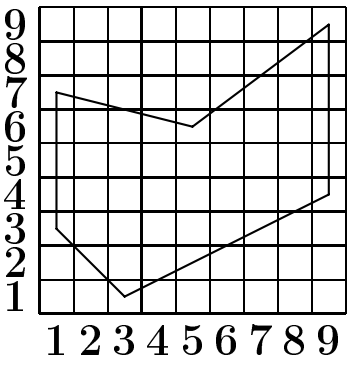
7. В цикле от Y_tek до Y_след:

   - выбрать из списка активных ребер и отсортировать X-координаты пересечений активных ребер со строкой сканирования;
   - определить интервалы и выполнить закраску;
   - перевычислить координаты пересечений для следующей строки сканирования.

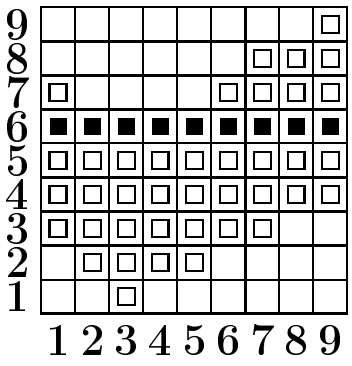
8. Проверить не достигли ли максимальной Y-координаты. Если достигли, то заливка закончена, иначе выполнить пункт ??.

9. Очистить список активных ребер от ребер, закончившихся на строке Y_след и перейти к пункту 4.

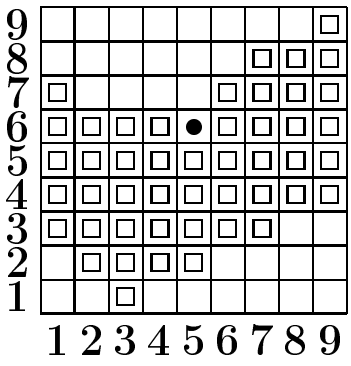
Результаты работы



а) Контур многоугольника



б) Заполнение процедурой V_FP0



в) Заполнение процедурой V_FP1

⊡ — пиксел занесен 1 раз; ⊙ — пиксел занесен дважды;

■ — пиксел занесен трижды;

Заливка области с затравкой

Задаются:

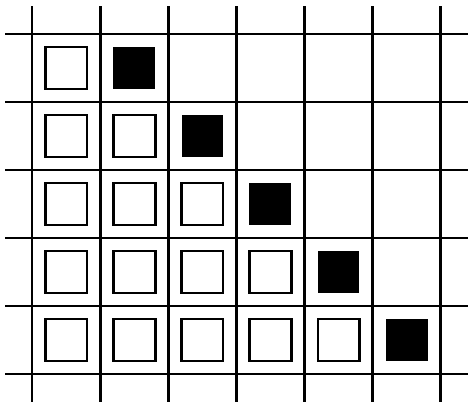
- заливаемая (перекрашиваемая) область,
- код пиксела, которым будет выполняться заливка,
- начальная точка в области, начиная с которой начнется заливка.

По способу задания области делятся на 2 типа:

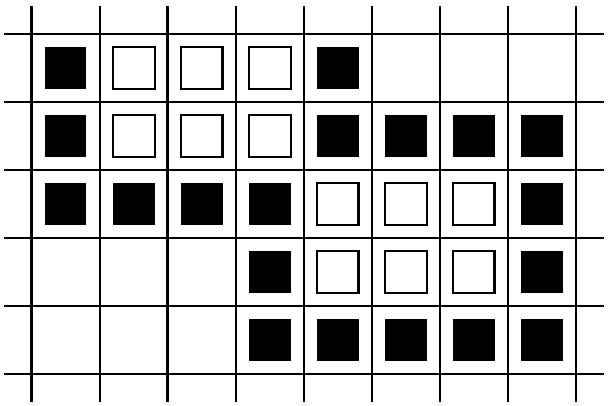
- гранично-определенные, задаваемые своей (замкнутой) границей, нарисованной определенным кодом пиксела.
- внутренне-определенные, нарисованные одним определенным кодом пиксела.

По способам доступа к соседним пикселам области делятся на следующие два типа:

ь	Область	Граница	Алгоритм заливки
1.	4-х связная	8-ми связная	4-х и 8-ми связный
2.	8-ми связная	4-х связная	8-ми связный



а) 4-х



б) 8-ми

связные, внутренне-определенные области

□ — пикселы области; ■ — пикселы границы.

Алгоритмы заливка области

Простой алгоритм

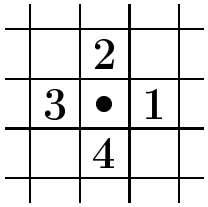
Занести координаты затравочного пиксела в стек;

While стек не пуст

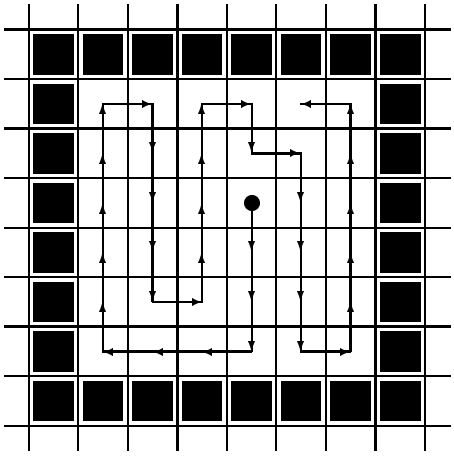
- Взять координаты пиксела из стека;
- Перекрасить пиксел;
- Проверить соседние пикселы;
- Если они не закрашены и не граничные то занести их координаты в стек;

Endwhile

Заливка 4-х связной области итеративным алгоритмом



а)
Порядок перебора соседних пикселов



б)
Порядок заливки области

Показатели алгоритма (стек 64 K):

- рекурсивная реализация — область $\leq 57 \times 57$
- итеративная реализация — область $\leq 110 \times 110$

Построчный алгоритм

Использует пространственную когерентность:

- пикселы в строке меняются только на границах;
- при перемещении к следующей строке размер заливаемой строки скорее всего или неизменен или меняется на 1 пиксел.

Занести координаты затравочного пиксела в стек;

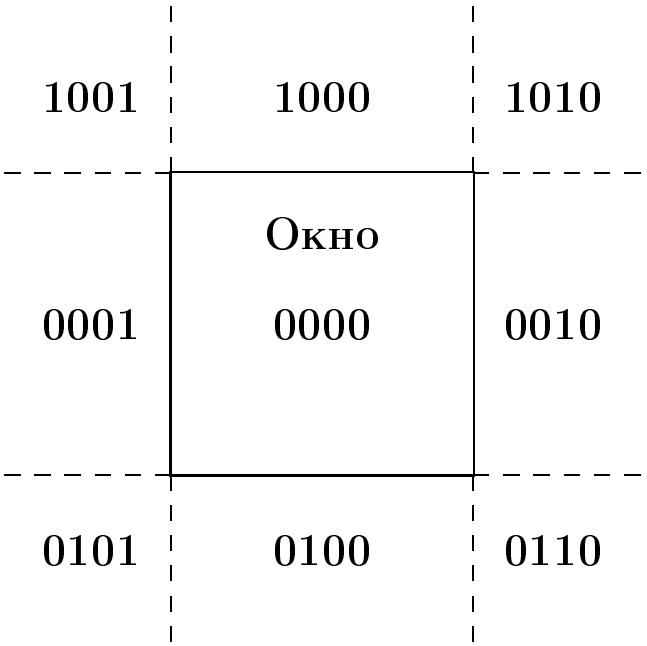
While стек не пуст

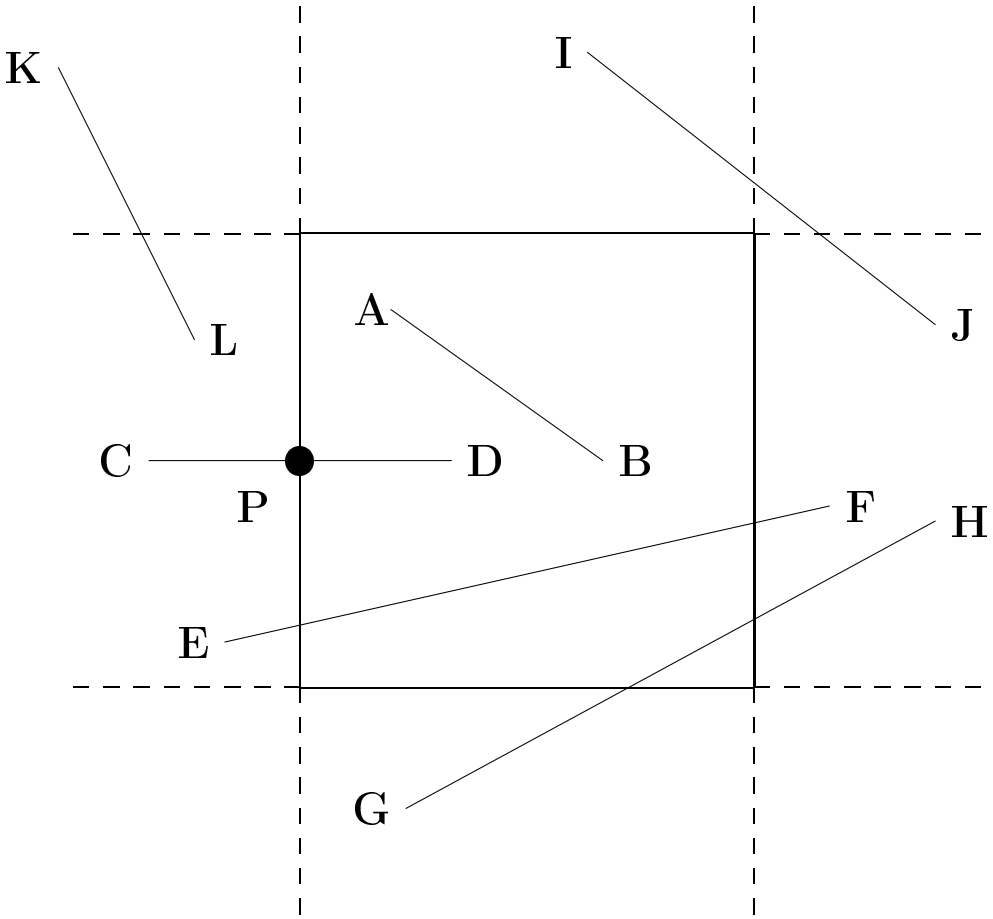
- Взять координаты затравки из стека;
- Перекрасить пиксел;
- Закрасить строку вправо и влево от затравки пока не встретится уже закрашенный или граничный пиксел;
- Запомнить координаты крайних закрашенных пикселов $X_{лев}$ и $X_{прав}$;
- В диапазоне от $X_{лев}$ до $X_{прав}$ в выше и нижележащей строке отыскиваются крайние правые пикселы в еще незакрашенных сегментах.
- Координаты отысканных пикселов запоминаются в стеке как затравки.

Endwhile

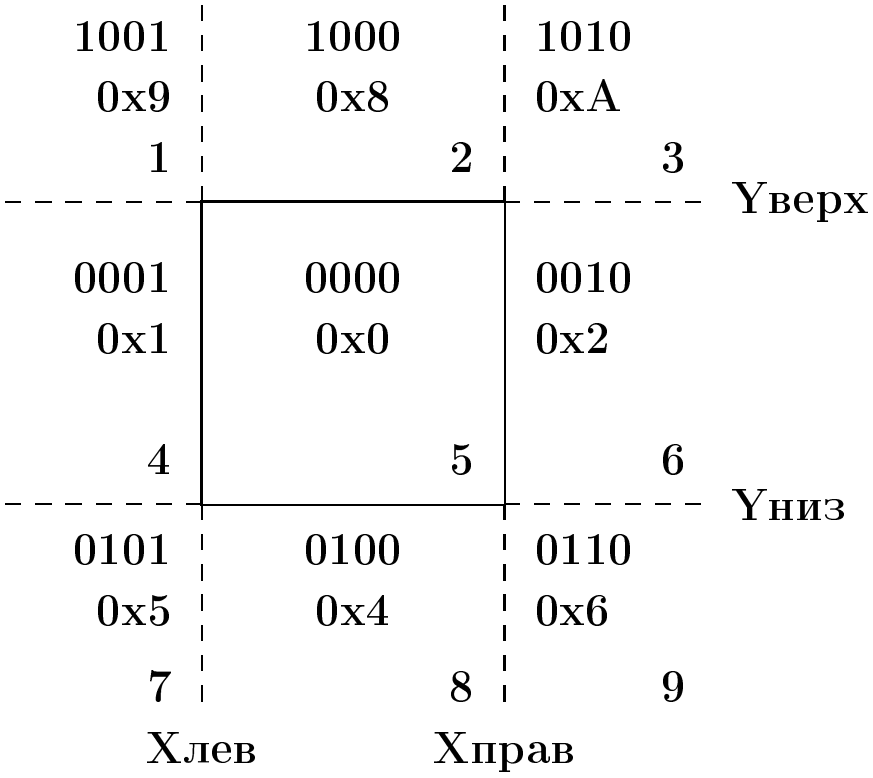
Отсечение отрезков

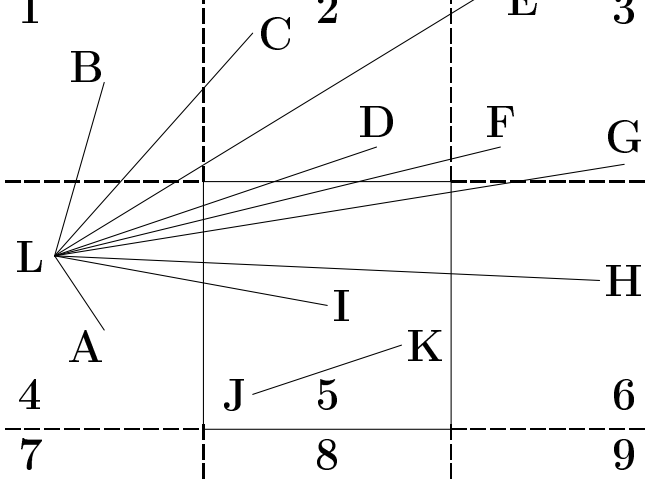
Алгоритм Коэна-Сазерленда

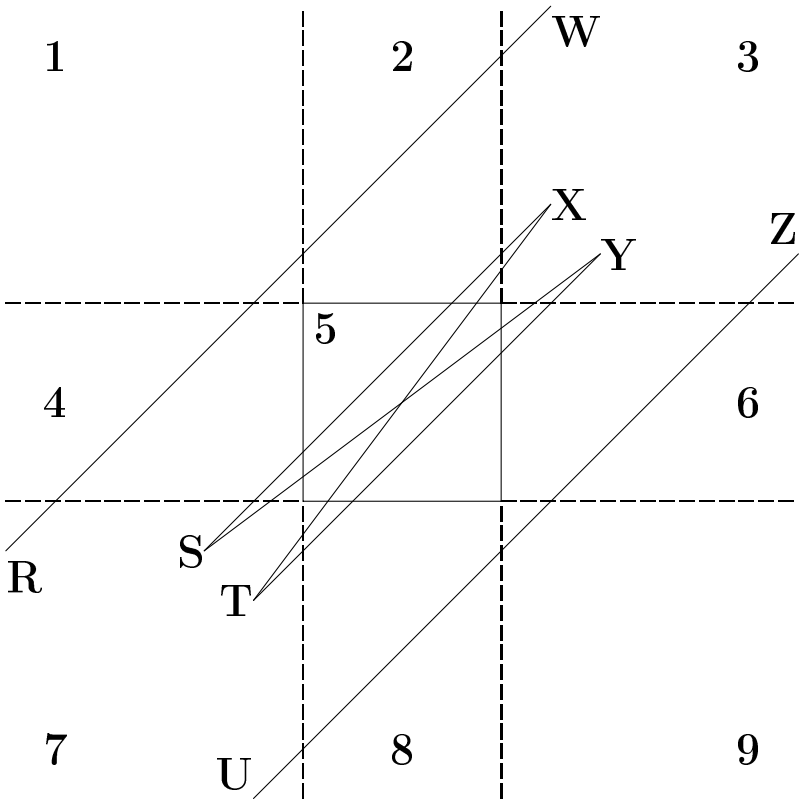




FC-алгоритм

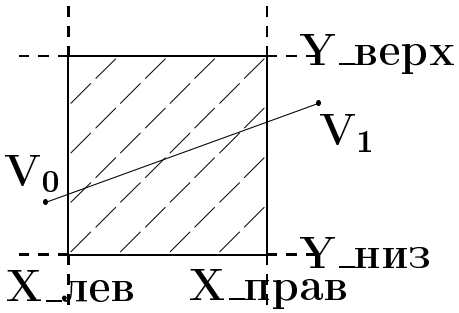

1001
1000
1010
0x9
0x8
0xA
1
2
3
Yверх
0001
0000
0010
0x1
0x0
0x2
4
5
6
Yниз
0101
0100
0110
0x5
0x4
0x6
7
8
9
Xлев
Xправ


1
2
3
C
B
D
F
G
L
H
I
A
K
4
J
5
6
7
8
9

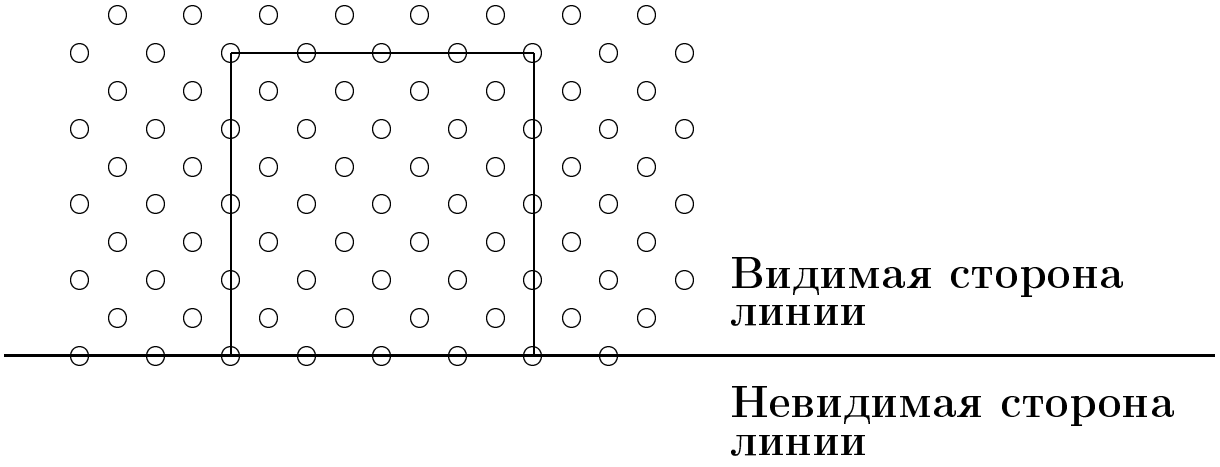

1
2
W
3
X
Z
Y
5
4
6
S
R
T
7
U
8
9

Двумерный алгоритм Лианга-Барски

(Liang-Barsky, LB-алгоритм)



$$X \le x \le X$$
$$Y \le y \le Y$$



Алгоритм Лианга-Барски. Пример расчета отсечения

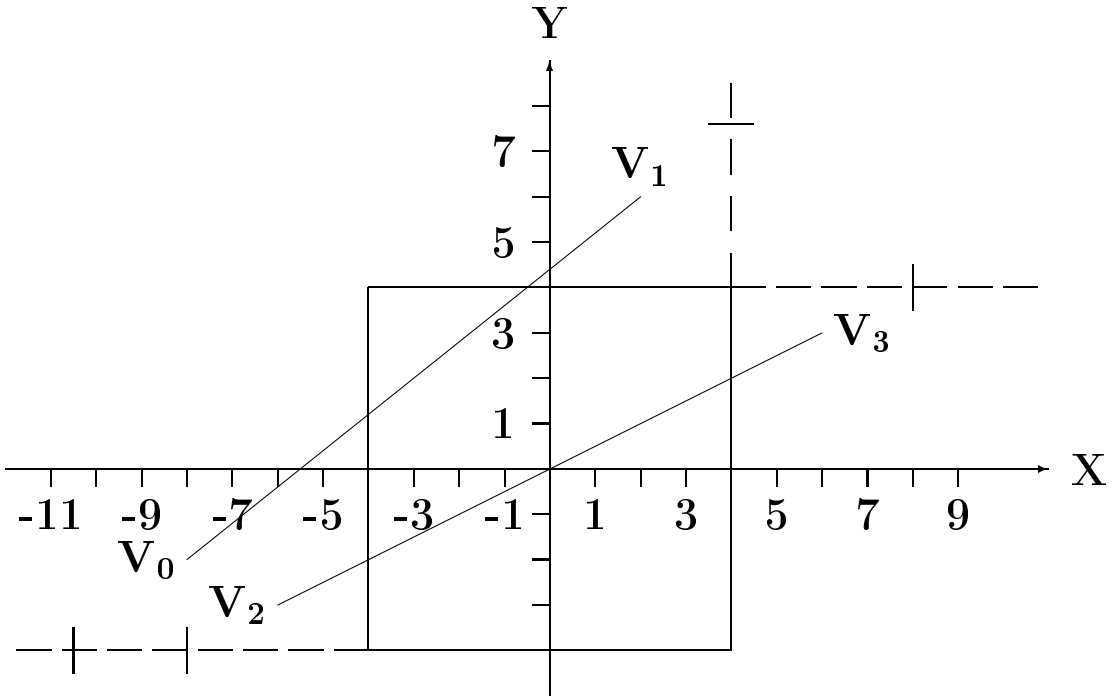
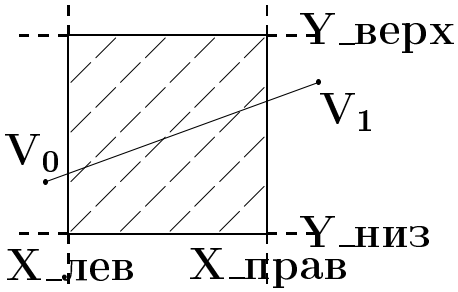


Таблица 1: Расчет отсечения для отрезка V_0V_1

	Ребро окна	t	t0, t1	X	Y
Внутрь	Левое	4/10	t0= tmax= 4/10	-4	1.2
	Нижнее	-2/8	вне [0,1]		
Наружу	Правое	12/10	вне [0,1]		
	Верхнее	6/8	t1= tmin= 6/8	-0.5	4

Таблица 2: Расчет отсечения для отрезка V_2V_3

	Ребро окна	t	t0, t1	X	Y
Внутрь	Левое	2/12	t0= tmax= 2/12	-4	-2
	Нижнее	-1/6	вне [0,1]		
Наружу	Правое	10/12	t1= tmin=10/12		
	Верхнее	7/6	вне [0,1]	4	2



$$X \le x \le X$$
$$Y \le y \le Y$$

Параметрическое уравнение прямой V_0V_1

$$X(t) = X_0 + dX \cdot t; \qquad dX = X_1 - X_0; \qquad 0 \le t \le 1;$$
$$Y(t) = Y_0 + dY \cdot t; \qquad dY = Y_1 - Y_0;$$

$$V(t) = V_0 + dV \cdot t; \qquad dV = V_1 - V_0.$$

$-\infty \le t \le +\infty$ — удлиненная прямая

Подставим уравнение прямой в условие нахождения в окне. Равенства — границы окна. Неравенства — внутри окна.

$$\begin{array}{ccccc} X & \le & X_0 + dX \cdot t & \le & X \\ Y & \le & Y_0 + dY \cdot t & \le & Y \end{array} \Longrightarrow$$

$$\begin{array}{ccccc} -dX \cdot t & \le & X_0 - X; & \text{и} & dX \cdot t \le X - X_0; \\ -dY \cdot t & \le & Y_0 - Y; & \text{и} & dY \cdot t \le Y - Y_0. \end{array} \Longrightarrow$$

$$t \cdot P_i \le Q_i; \quad i = 1,2,3,4 \quad \Longrightarrow \quad t \le Q_i/P_i.$$

$$\begin{array}{ll} P_1 = -dX; & Q_1 = X_0 - X; \\ P_2 = dX; & Q_2 = X - X_0; \\ P_3 = -dY; & Q_3 = Y_0 - Y; \\ P_4 = dY; & Q_4 = Y - Y_0. \end{array}$$

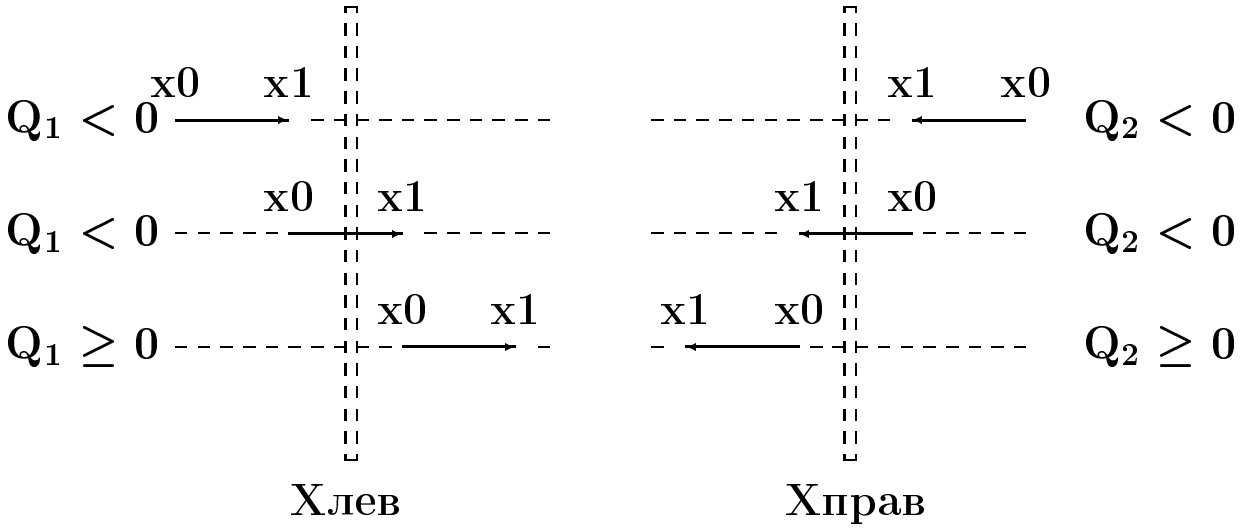
$Q_i \ge 0 \Longrightarrow V_0$ на видимой стороне границы
$Q_i < 0 \Longrightarrow V_0$ на невидимой стороне границы

Решаем $t \le Q_i/P_i$. 1) $P_i < 0$; 2) $P_i > 0$; 3) $P_i = 0$.

1) $P_i < 0 \Longrightarrow t \ge Q_i / P_i$. Снаружи внутрь.

$Q_1 = x0 - X$ $\quad$ $Q_2 = X - x0$

$P_1 = x0 - x1 < 0$ $\quad$ $P_2 = x1 - x0 < 0$



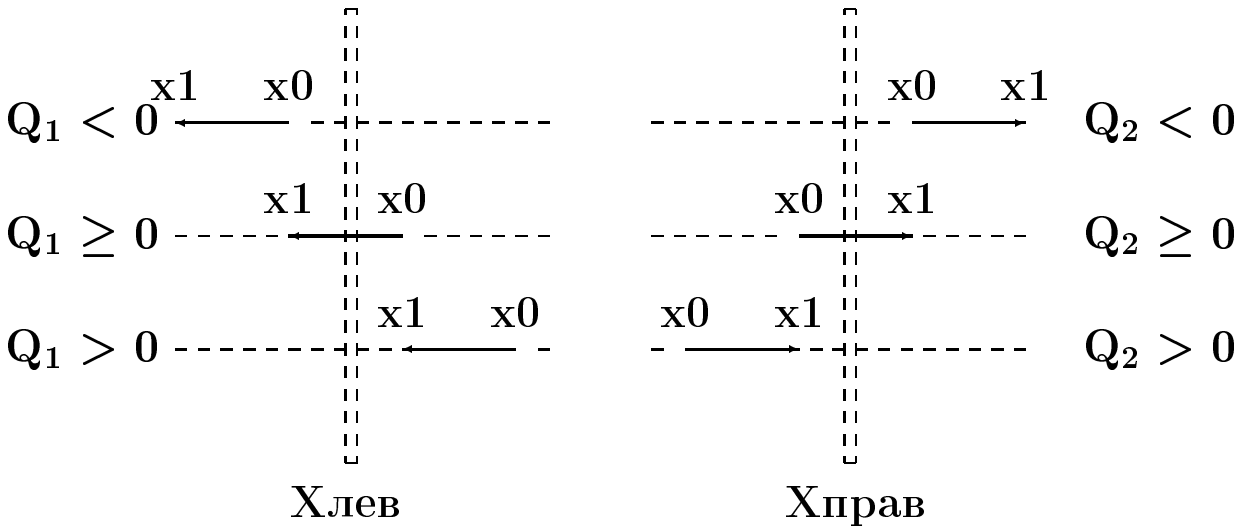
$t = Q_i/P_i$ на границе,

$t > Q_i/P_i$ внутри окна $\Longrightarrow$ на границе минимум t.

2) $P_i > 0 \Longrightarrow t \le Q_i / P_i$. Изнутри наружу.

$Q_1 = x0 - X$ $\quad$ $Q_2 = X - x0$

$P_1 = x0 - x1 > 0$ $\quad$ $P_2 = x1 - x0 > 0$

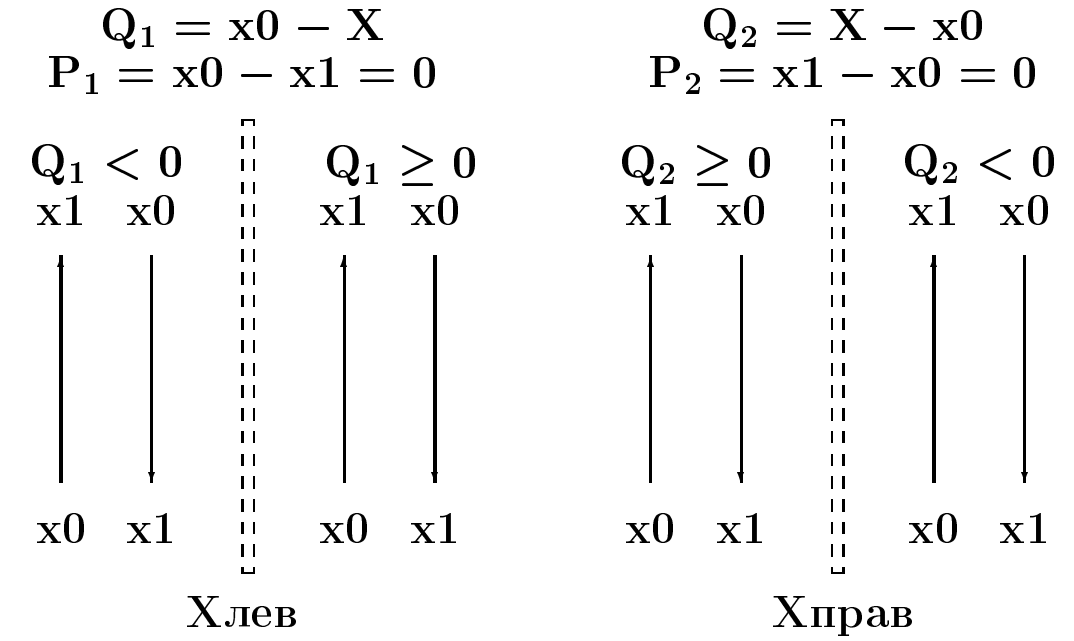


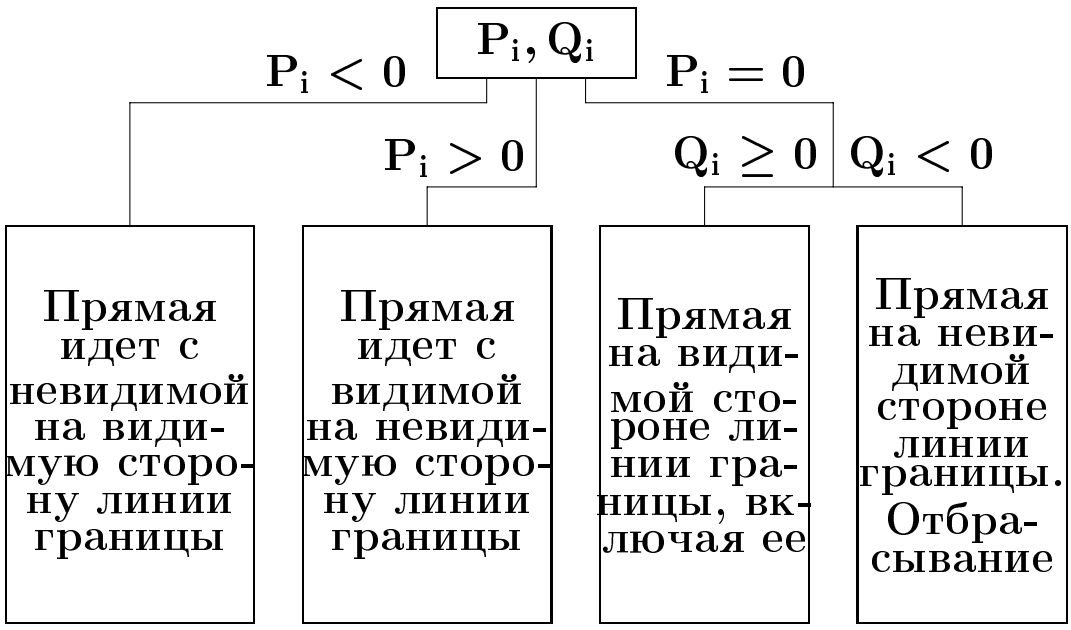
$t = Q_i/P_i$ на границе,

$t < Q_i/P_i$ внутри окна $\Longrightarrow$ на границе максимум t.

3) $P_i = 0 \Longrightarrow 0 \geq Q_i / P_i$. Параллельно стороне.

Есть решение $0 \geq Q_i / P_i$ при $Q_i \geq 0$ и нет при $Q_i < 0$.





$$t_0 \geq \max(\{Q_i/P_i | P_i < 0,\ i = 1,2,3,4\} \cup \{0\}),$$
$$t_1 \leq \min(\{Q_i/P_i | P_i > 0,\ i = 1,2,3,4\} \cup \{1\}).$$

$$t_0 \leq t \leq t_1.$$

Двумерный алгоритм Кируса-Бека

(Curus-Beck, CB-алгоритм)

Отсечение выпуклым многоугольником.

Ориентация отрезка и положение точки относительно окна определяются скалярным произведением вектора внутренней нормали $\vec{N}_i$ к стороне окна на искомый вектор.

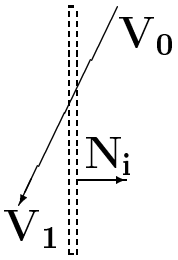
Направленность отрезка $\overrightarrow{V_0V_1}$:

$$P_i = \vec{N}_i \cdot \vec{P} = \vec{N}_i \cdot (\vec{V}_1 - \vec{V}_0).$$

$P_i < 0$ — $\overrightarrow{V_0V_1}$ идет с изнутри наружу

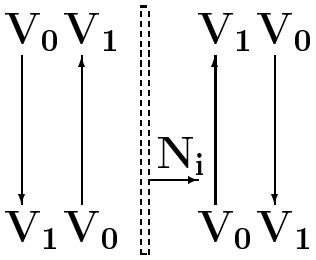
$P_i = 0$ — $\overrightarrow{V_0V_1} \parallel$ стороне окна или же точка $V_0 = V_1$

$P_i > 0$ — $\overrightarrow{V_0V_1}$ идет снаружи внутрь



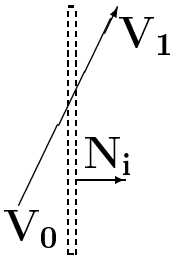
а)

Изнутри наружу



б)

Параллельно границе



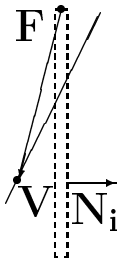
в)

Снаружи внутрь

Положение точки V на отрезке V_0V_1 относительно окна:

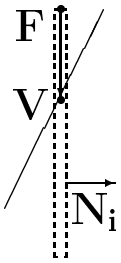
$$Q_i = \vec{N}_i \cdot \vec{Q} = \vec{N}_i \cdot [\vec{V}(t) - \vec{F}_i] \qquad i = 1, 2, 3 \ldots$$

$$Q_i < 0 \quad V(t)$$
$$Q_i = 0 \quad V(t)$$
$$Q_i > 0 \quad V(t)$$



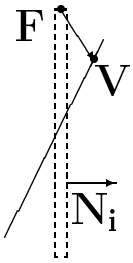
а)

Точка V вне



б)

Точка V на границе



в)

Точка V внутри

$$V(t) = V_0 + (V_1 - V_0) \cdot t; \quad 0 \geq t \geq 1 \implies$$

Условие пересечения $\overrightarrow{V_0V_1}$ с границей:

$$\vec{N}_i \cdot [\vec{V}(t) - \vec{F}_i] = \vec{N}_i \cdot [\vec{V}_0 + (\vec{V}_1 - \vec{V}_0) \cdot t - \vec{F}_i] = 0$$

или

$$(\vec{N}_i \cdot \vec{P}) \cdot t + \vec{N}_i \cdot \vec{Q} = P_i \cdot t + Q_i = 0 \implies$$

$$t = -Q_i \,/\, P_i = -(\vec{N}_i \cdot \vec{Q}) \,/\, (\vec{N}_i \cdot \vec{P})$$

$$P_i \neq 0, \; i = 1, 2, 3, \ldots$$

Итак:

$t = -Q_i / P_i \quad P_i \neq 0,\ i = 1, 2, 3, \ldots$

$P_i < 0$ — $\overrightarrow{V_0V_1}$ идет с изнутри наружу

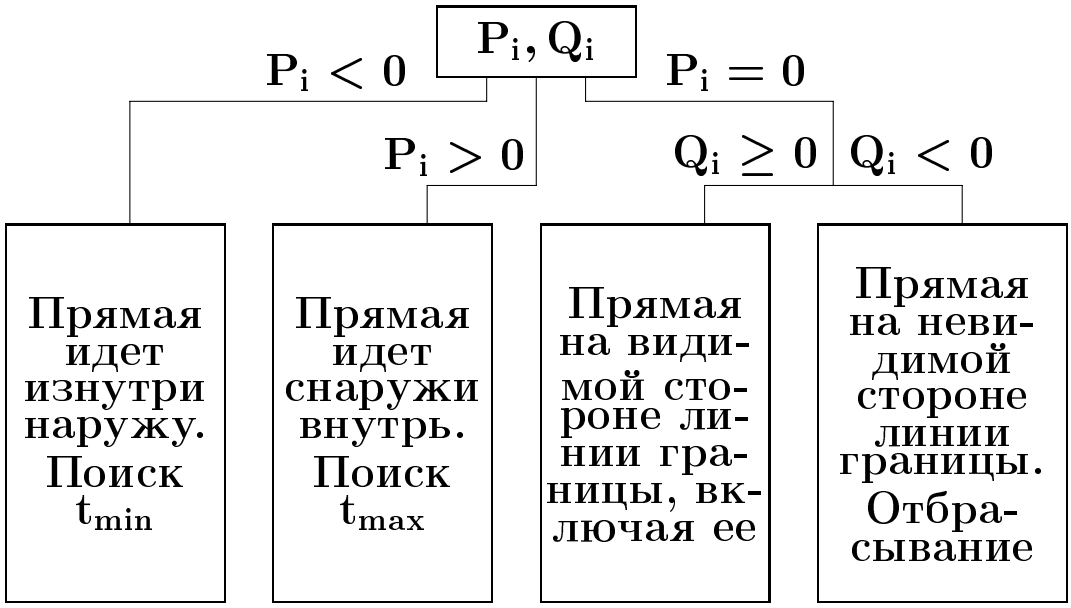
$P_i = 0$ — $\overrightarrow{V_0V_1} \parallel$ стороне окна или же точка $V_0 = V_1$

$P_i > 0$ — $\overrightarrow{V_0V_1}$ идет снаружи внутрь

$Q_i < 0$ — точка $V(t)$ на внешней стороне

$Q_i = 0$ — точка $V(t)$ на границе

$Q_i > 0$ — точка $V(t)$ на внутренней стороне



$$t_0 \geq \max(\{-Q_i/P_i | P_i > 0,\ i = 1, 2, \ldots\} \cup \{0\}),$$
$$t_1 \leq \min(\{-Q_i/P_i | P_i < 0,\ i = 1, 2, \ldots\} \cup \{1\}).$$

Проверка выпуклости и определение нормалей

Проверкой знаков векторных произведений смежных ребер

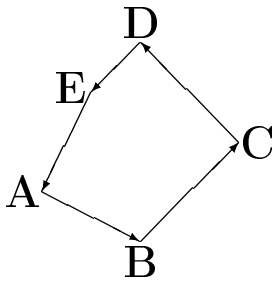
$$|\vec{A} \times \vec{B}| = A \cdot B \cdot \sin\left(\overset{\wedge}{AB}\right)$$

Если все знаки $= 0$, то многоугольник вырожден в отрезок

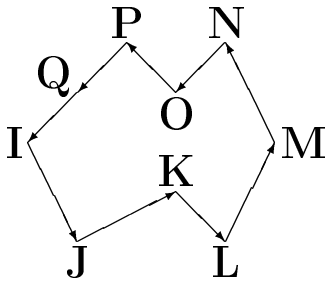
Если все знаки неположительные, то многоугольник выпуклый с обходом вершин по часовой стрелке. Направление внутренней нормали к стороне — поворот стороны на $+90°$.

Если все знаки неотрицательные, то многоугольник выпуклый с обходом вершин против часовой стрелки. Направление внутренней нормали к стороне — поворот стороны на $-90°$.

Если все знаки различны, то многоугольник невыпуклый.



а) выпуклый

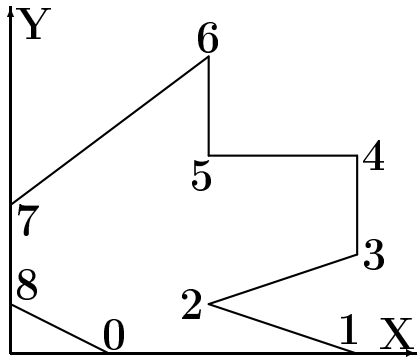


б) невыпуклый

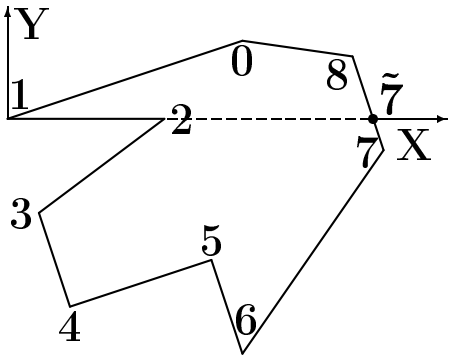
Проверка выпуклости и разбиение окна методом переносов и поворотов

Алгоритм метода при обходе вершин многоугольника против часовой стрелки:

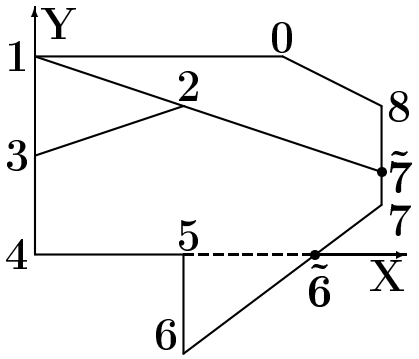
1. Сдвиг многоугольника для переноса i-й вершины в начало координат.
2. Поворот многоугольника для совмещения (i+1)-й вершины с +X полуосью.
3. Анализ знака Y-координаты (i+2)-й вершины.
   Если $Y_{i+2} \geq 0$, то в (i+1)-й вершине выпуклость.
   Если $Y_{i+2} \geq 0$, то в (i+1)-й вершине невыпуклость.
   Многоугольник разрезается на два вдоль +X-полуоси.
   Для этого вычисляется пересечение +X-полуоси с первой из сторон.
   1-й многоугольник — вершины с (i+1)-й до точки пересечения — вершины 2, 3, 4, 6, 7, $\tilde{7}$ на рис. б)
   2-й многоугольник — все остальные вершины — вершины $\tilde{7}$, 8, 0, 1 на рис. б)



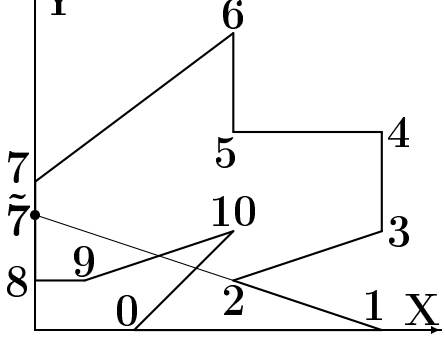
а)
Исходное окно



б)
Невыпуклость после вершины 2



в)
Невыпуклость после вершины 5

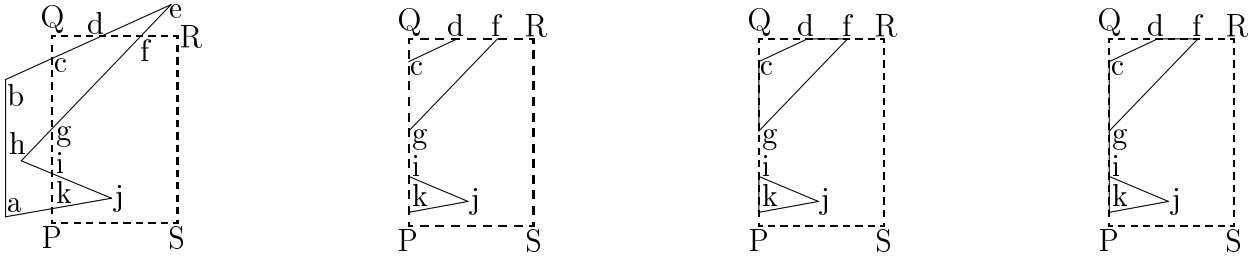


На окнах с самопересечением сторон алгоритм не работает

1 Отсечение многоугольника

Многоугольники особенно важны в растровой графике как средство задания поверхностей.

Рассмотренные в предыдущем разделе алгоритмы отсечения линий должны быть модифицированы для того, чтобы можно было их использовать для отсечения многоугольников. На рис. ?? а) показаны окно отсечения PQRS и исходный многоугольник abehj. В результате применения алгоритмов отсечения линий мы получим набор несвязанных между собой отрезков cd, fg, ij, jk — рис. ?? б). Желаемый же результат отсечения, необходимый при закраске отсеченного многоугольника, представлен на рис. ?? в). Видно, что в состав отсеченного многоугольника должны войти фрагменты границ окна отсечения — ребра cg, df, ki.



а) До отсечения

б) Отсечение отрезков

в) Желаемый результат

г) Замыкание отрезков

Рис. 1: Отсечение многоугольника

Казалось бы, задачу формирования многоугольника можно решить, соединяя предыдущую и последующие точки пересечения ребер многоугольника с окном. В этом случае воможно возникновение паразитных ребер результирующего многоугольника, идущих вдоль окна, — ребро ig на рис. ?? г).

В ряде случаев, например, при отсечении по границе экрана эти паразитные ребра несущественны. Рассмотренный подход позволяет правильно отсечь для простых случаев (рис. ?? а)) и дает ошибку, если отсекаемый многоугольник охватывает вершину окна (см. рис. ?? б)). В последнем случае результат отсечения — треугольник a, a_s, d_s, вместо прямоугольника a, a_s, R, d_s.

Рис. 2: Использование отсечения отрезков с замыканием

Таким образом, алгоритм отсечения многоугольника должен в результате отсечения давать один или несколько замкнутых многоугольников. При этом могут быть добавлены новые ребра, а имеющиеся или сохранены или разделены или даже отброшены. Существенно, чтобы границы окна, которые не ограничивают видимую часть отсекаемого многоугольника, не входили в состав результата отсечения. Если это не выполняется, то возможна излишняя закраска границ окна.

Далее мы рассмотрим три алгоритма корректно решающие задачу отсечения сплошного многоугольника. Первые два алгоритма более быстро работают, но генерируют лишние ребра вдоль границы окна, как это продемонстрировано на рис. ?? г). Последний алгоритм свободен от указанного недостатка.

В заключение рассмотрим алгоритм отсекающий произвольный многоугольник по произвольному же окну, но он может в результате пересечения дать незамкнутый многоугольник.

В общем, при отсечении многоугольников возникают два типа задач — отображение части изображения попавшей в окно и наоборот, отображение изображения, находящегося вне окна. Все здесь рассматриваемые алгоритмы могут использоваться в обоих случаях.

1.1 Алгоритм Сазерленда-Ходгмана

Простой метод решения проблемы охвата отсекаемым многоугольником вершины окна предлагается в алгоритме Сазерленда-Хогдмана [?], когда весь многоугольник последовательно отсекается каждой границей окна, как это показано на рис. ??.

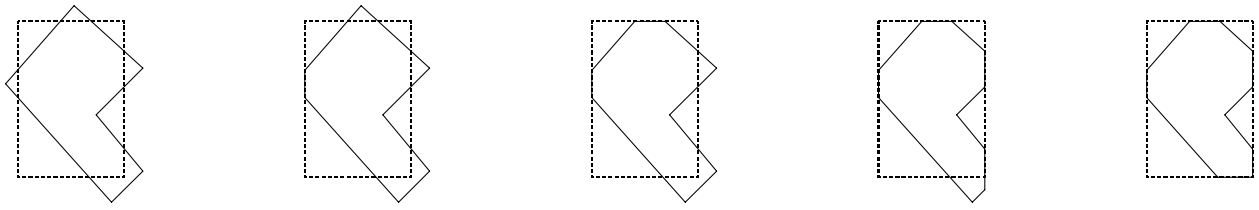


Рис. 3: Последовательное отсечение многоугольника сторонами окна.

При отсечении ребра, соединяющего очередную пару вершин, возможны 4 случая взаимного расположения (рис. ??):

Вариант	Расположение ребра	Результат
а)	ребро на внешней стороне границы	нет вывода
б)	ребро входит снаружи в окно	точки b_s и c
в)	ребро на внутренней стороне границы	точка d
г)	ребро выходит из окна наружу	точка d_s

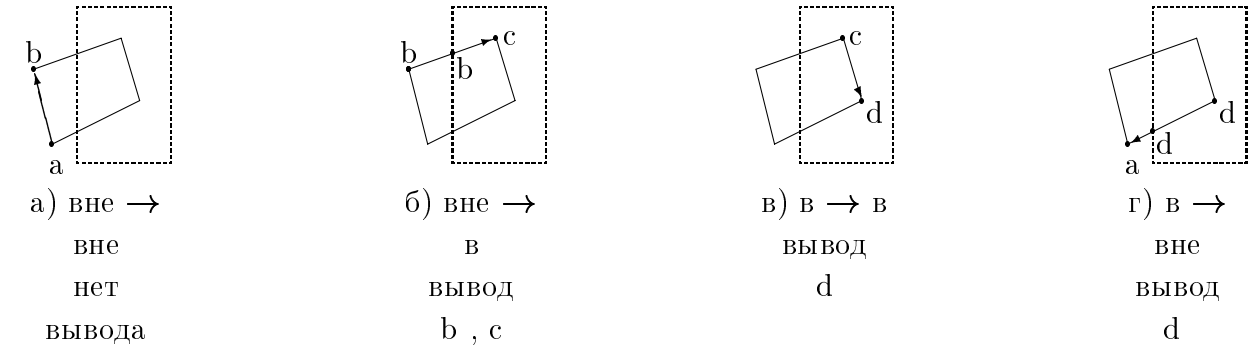
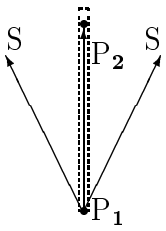


Рис. 4: Относительное расположение ребра и границы окна

Для определения взаимного расположения и направленности можно использовать анализ знака Z-компоненты векторного произведения вектора $\mathbf{P_1P_2}$, проведенного из начальной в конечную точку текущего ребра окна, на вектор $\mathbf{P_1S}$ из начальной точки текущего ребра окна в проверяемую вершину многоугольника (рис. ??).

Отсечение многоугольника последовательно относительно каждой из сторон окна приводит к необходимости запоминания вершин промежуточных многоугольников (см. рис. ??). На рис. ?? показано отсечение треугольника на прямоугольном окне и изменение списка вершин многоугольника при прохождении через после-



Если Z-компонента $\mathbf{P_1P_2} \times \mathbf{P_1S} < \mathbf{0}$,
то поворот от $\mathbf{P_1P_2}$ $\mathbf{P_1S}$ по часовой стрелке,
т.е. точка S внутри окна.

Если Z-компонента $\mathbf{P_1P_2} \times \mathbf{P_1S} > \mathbf{0}$,
то поворот от $\mathbf{P_1P_2}$ $\mathbf{P_1S}$ против часовой стрелки,
т.е. точка S вне окна.

Рис. 5: Определение взаимного расположения окна и вершины

довательные отсечения сторонами окна.

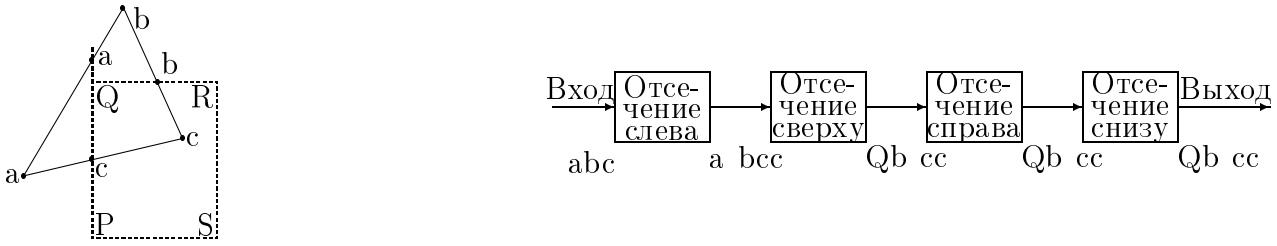


Рис. 6: Конвейер отсечения всего многоугольника сторонами окна.

Сазерленд и Хогдман предложили метод сокращения затрат на промежуточное хранение. Для этого, вместо отсечения всех вершин (ребер) относительно одной стороны окна, после отсечения вершины одной из сторон окна полученный результат отсекается относительно следующей стороны окна и т.д.

Предложена аппаратная реализация этого алгоритма, состоящая из четырех идентичных ступеней отсечения без промежуточной памяти [?].

В алгоритме Сазерленда-Ходгмана в результат могут заноситься границы окна, даже если они и не ограничивают видимую часть отсеченного многоугольника. Это можно устранить дополнительным анализом, либо используя более сложный алгоритм отсечения.

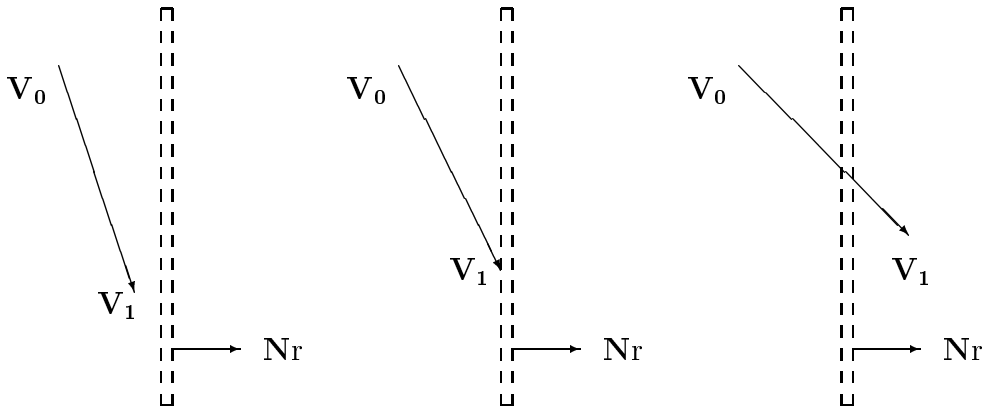
1.2 Простой алгоритм отсечения многоугольника

В данном разделе рассматривается простой алгоритм отсечения, который подобно алгоритму Сазерленда-Ходгмана может генерировать лишние стороны для отсеченного многоугольника, проходящие вдоль ребра окна отсечения. Но этот алгоритм несколько более быстрый и использует те же подпрограммы обработки многоугольного окна отсечения, что и алгоритм Кируса-Бека.

Многоугольник отсекается одним ребром выпуклого окна отсечения. В результате такого отсечения формируется новый многоугольник, который затем отсекается следующим ребром и т.д., пока не будет выполнено отсечение последним ребром окна.

Основная здесь процедура — процедура отсечения отдельным ребром, определяющая взаимное расположение очередной стороны многоугольника и ребра отсекателя и генерирующая соответствующие выходные данные.

Возможны 9 различных случаев расположения ребра окна и отсекаемой стороны, показанных на рис. ??–??.



Конечная точка: 1) вне 2) на ребре 3) внутри окна

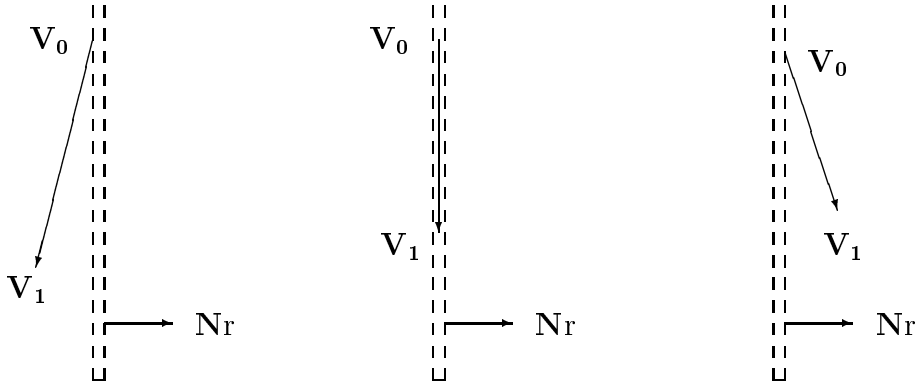
Рис. 7: Начальная точка вне ребра окна отсечения.

На них V_0 и V_1 — начальная и конечная точки отсекаемой стороны многоугольника, соответственно; Nr — нормаль к ребру окна отсечения, направленная внутрь окна.

Из этих рисунков очевидны правила генерации выходных данных, зависящие от варианта взаимного расположения:

1) Нет выходных данных.

2) В выходные данные заносится конечная точка.



Конечная точка: 4) вне 5) на ребре 6) внутри окна

Рис. 8: Начальная точка на ребре окна отсечения.

3) Рассчитывается пересечение и в выходные данные заносятся точка пересечения и конечная точка.

4) Нет выходных данных.

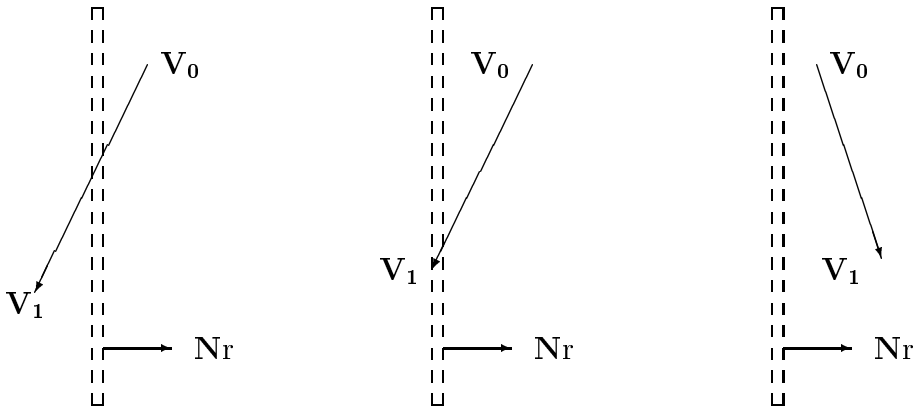
5) В выходные данные заносится конечная точка.

6) В выходные данные заносится конечная точка.

7) Рассчитывается пересечение и в выходные данные заносится только точка пересечения.

8) В выходные данные заносится конечная точка.

9) В выходные данные заносится конечная точка.



Конечная точка: 7) вне 8) на ребре 9) внутри окна

Рис. 9: Начальная точка внутри окна отсечения.

Для определения взаимного расположения, подобно алгоритму отсечения Кируса-Бека, используется ска-

лярное произведение Q вектора нормали на вектор, проведенный из начала ребра в анализируемую точку. Пояснение см. на рис. 8.10.

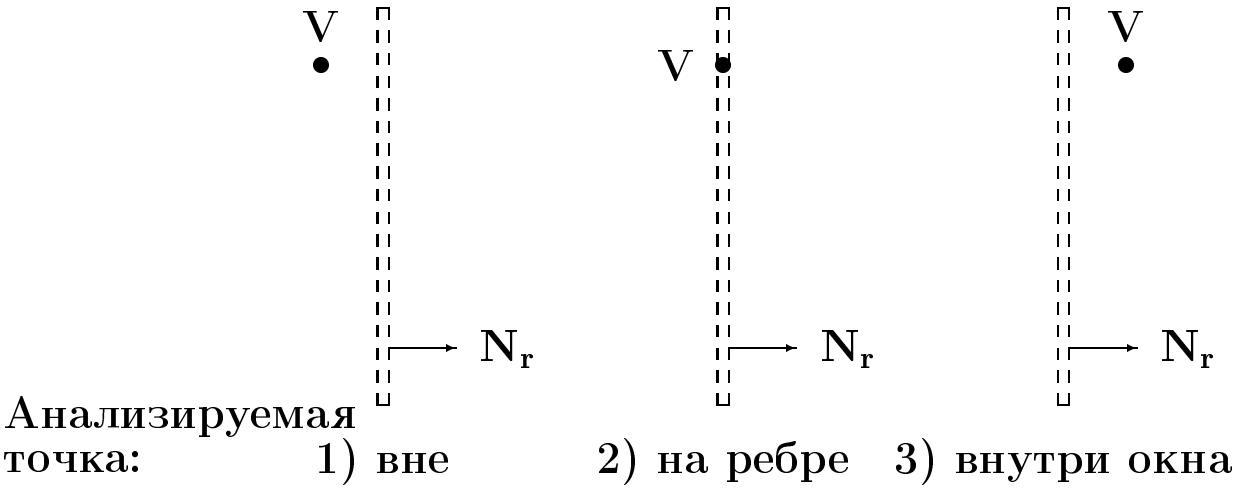


Рис. 8.10. Анализ расположения точки относительно ребра окна отсечения.

Таким образом, для определения взаимного расположения начальной V_0 и конечной V_1 точек отсекаемой стороны и ребра отсечения с вектором его начала R, надо вычислить:

$$Qn = (V_0 - R) \cdot Nr$$
$$Qk = (V_1 - R) \cdot Nr.$$

Расчет пересечения, если он требуется, производится аналогично алгоритму Кируса-Бека с использованием параметрического представления линии:

$$V(t) = V_0 + (V_1 - V_0) \cdot t.$$

Вначале находится значение параметра t для точки пересечения по формуле (см. описание алгоритма Кируса-Бека):

$$t = -Qn / Pn,$$

где Qn — скалярное произведение вектора нормали к ребру окна на вектор из начала ребра в начальную точку стороны, уже вычисленное при определении расположения начальной точки, а $\mathbf{Pn} = (\mathbf{V_1} - \mathbf{V_0}) \cdot \mathbf{Nr}$ — скалярное произведение вектора нормали к ребру окна на вектор из начальной в конечную точки отсекаемой стороны.

Легко выразить это произведение через уже вычисленные величины Qn и Qk:

$$\begin{aligned}\mathbf{Pn} &= (\mathbf{V_1} - \mathbf{V_0}) \cdot \mathbf{Nr} \\ &= (\mathbf{V_1} - \mathbf{V_0} - \mathbf{R} + \mathbf{R}) \cdot \mathbf{Nr} \\ &= (\mathbf{V_1} - \mathbf{R}) \cdot \mathbf{Nr} - (\mathbf{V_0} - \mathbf{R}) \cdot \mathbf{Nr} \\ &= \mathbf{Qk} - \mathbf{Qn}.\end{aligned}$$

Таким образом, точке пересечения соответствует значение параметра t, равное:

$$\mathbf{t} = \mathbf{Qn} \,/\, (\,\mathbf{Qn} - \mathbf{Qk}\,).$$

Значения координат пересечения находятся из:

$$\mathbf{Xp} = \mathbf{X_0} + (\mathbf{X_1} - \mathbf{X_0}) \cdot \mathbf{t}; \quad \mathbf{Yp} = \mathbf{Y_0} + (\mathbf{Y_1} - \mathbf{Y_0}) \cdot \mathbf{t}.$$

Описанный алгоритм реализован в процедуре V_Plclip, приведенной в Приложении 8.

1.3 Алгоритм отсечения многоугольника Вейлера-Азертона

В предыдущих разделах были рассмотрены два алгоритма отсечения многоугольника, последовательно отсекающие произвольный (как выпуклый, так и невыпуклый) многоугольник каждой из сторон выпуклого

окна. Зачастую же требуется отсечение по невыпуклому окну. Кроме того оба рассмотренных алгоритма могут генерировать лишние стороны для отсеченного многоугольника, проходящие вдоль ребра окна отсечения. Далее рассматриваемый алгоритм Вейлера-Азертона [?, ?, Род89] свободен от указанных недостатков ценой заметно большей сложности и меньшей скорости работы.

Предполагается, что каждый из многоугольников задан списком вершин, причем таким образом, что при движении по списку вершин в порядке их задания внутренняя область многоугольника находится справа от границы.

В случае пересечения границ и отсекаемого многоугольника и окна возникают точки двух типов:

- входные точки, когда ориентированное ребро отсекаемого многоугольника входит в окно,
- выходные точки, когда ребро отсекаемого многоугольника идет с внутренней на внешнюю стороны окна.

Общая схема алгоритма Вейлера-Азертона для определения части отсекаемого многоугольника, попавшей в окно, следующая:

1. Строятся списки вершин отсекаемого многоугольника и окна.

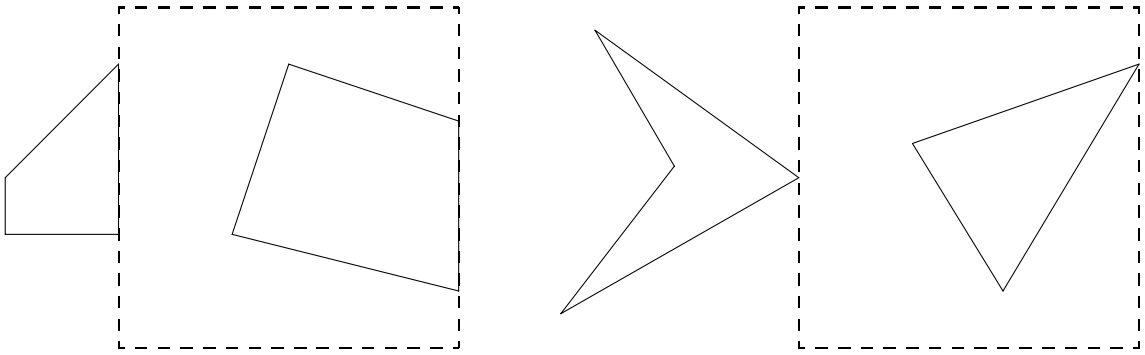
2. Отыскиваются все точки пересечения. При этом расчете касания не считаются пересечением, т.е. когда вершина или ребро отсекаемого многоугольника инцидентна или совпадает со стороной окна (рис. ?? и ??).

3. Списки координат вершин отсекаемого многоугольника и окна дополняются новыми вершинами — ко-

ординатами точек пересечения. Причем если точка пересечения Pk находится на ребре, соединяющем вершины Vi, Vj, то последовательность точек Vi, Vj превращается в последовательность Vi, Pk, Vj. При этом устанавливаются двухсторонние связи между одноименными точками пересечения в списках вершин отсекаемого многоугольника и окна.
Входные и выходные точки пересечения образуют отдельные подсписки входных и выходных точек в списках вершин.

4. Определение части обрабатываемого многоугольника, попавшей в окно выполняется следующим образом:
Если не исчерпан список входных точек пересечения, то выбираем очередную входную точку.
Двигаемся по вершинам отсекаемого многоугольника пока не обнаружится следующая точка пересечения; все пройденные точки, не включая прервавшую просмотр, заносим в результат; используя двухстороннюю связь точек пересечения, переключаемся на просмотр списка вершин окна.
Двигаемся по вершинам окна до обнаружения следующей точки пересечения; все пройденные точки, не включая последнюю, прервавшую просмотр, заносим в результат.
Используя двухстороннюю связь точек пересечения, переключаемся на список вершин обрабатываемого многоугольника.
Эти действия повторяем пока не будет достигнута исходная вершина — очередная часть отсекаемого

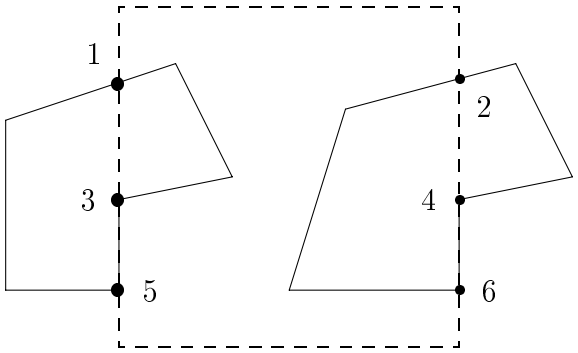
многоугольника, попавшая в окно, замкнулась. Переходим на выбор следующей входной точки в списке отсекаемого многоугольника.



Ребро отсекаемого многоугольника совпадает с окном.

Вершина отсекаемого многоугольника касается окна.

Рис. 10: Случаи не считающиеся пересечением.



Точки 1–4 считаются пересечением.

Точки 5,6 не считаются пересечением.

Рис. 11: Частные случаи пересечения.

Модификация этого алгоритма для определения части отсекаемого многоугольника, находящейся вне окна, заключается в следующем:

- **исходная точка пересечения пересечения берется из списка выходных точек,**
- **движение по списку вершин окна выполняется в обратном порядке, т.е. так чтобы внутренняя часть отсекателя была слева.**

На рис. ?? иллюстрируется отсечение многоугольника ABCDEFGHI окном PQRS по алгоритму Вейлера-Азертона.

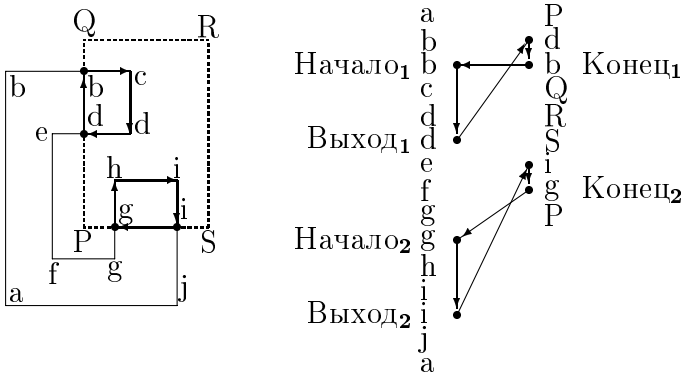


Рис. 12: Отсечение по алгоритму Вейлера-Азертона.

1.4 Алгоритм отсечения произвольных многоугольников

В начале раздела рассматривалась возможность использования алгоритмов отсечения отрезков для отсечения многоугольников. Проблема заключалась в том, что для построения замкнутого многоугольника требовалось соединение соседних точек пересечения многоугольника с окном, что не всегда приводит к правильному результату (см. рис. ?? г) и рис. ?? б)).

Далее рассматривается алгоритм, свободный от указанного недостатка, но при использовании данного алгоритма координаты ребер, сходящихся в некоторой вершине возможно будут отличаться на 1 пиксел.

На рис. ?? а) показано состояние до отсечения. Окно — вершины PQRS. Отсекаемый многоугольник — вершины abcd.

Отсечение выполняется в два этапа:

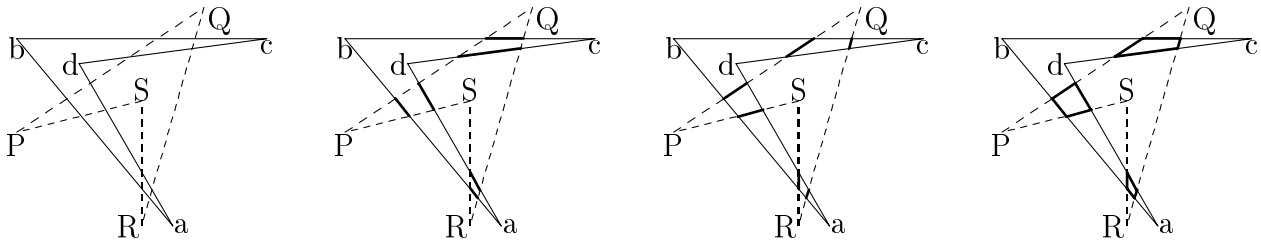
1. Отсечение многоугольника по окну.

   Рассчитываются все точки пересечения очередного ребра многоугольника со сторонами окна.

Соседние пары точек пересечений — результат отсечения. Они показаны на рис. ?? б) жирными линиями.

2. Многоугольник и окно меняются местами.
   Т.е. производится отсечение окна по многоугольнику. (см. рис. ?? в)).

Результат отсечения показан на рис. ?? г).



а) До отсечения

б) Отсечение многоугольника

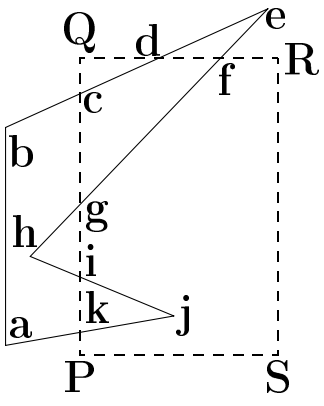
в) Отсечение окна

г) Результат отсечения

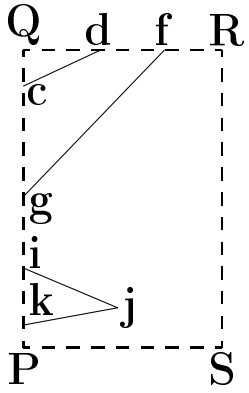
Рис. 13: Отсечение произвольных многоугольников.

Отсечение многоугольников

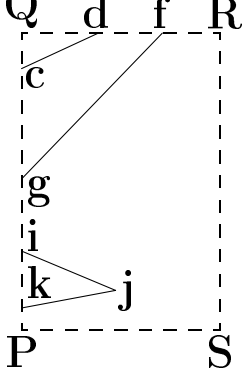
- отсекатель
- отсекаемый многоугольник



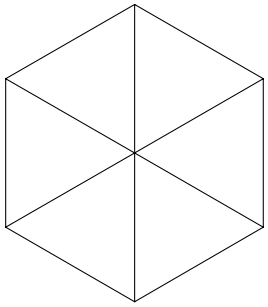
а) Корректное отсечение



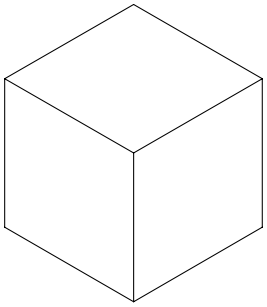
б) Неверное отсечение — в результат включены ребра Pd_s и Pg_s


Q
d
f
R
c
g
i
k
j
P
S

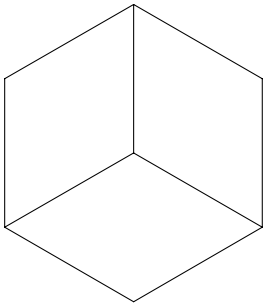
Удаление скрытых линий и поверхностей



а)
Изометрия



б)
Вид сверху



в)
Вид снизу

Классификация алгоритмов удаления:

1. По выбору удаляемых частей:
- невидимых линий, ребер, поверхностей, объемов.

2. По порядку обработки элементов сцены:
- удаление в произвольном порядке,
- в порядке, определяемом процессом визуализации.

3. По системе координат:
- алгоритмы работающие в пространстве объектов, когда каждая из N граней объекта сравнивается с остальными N-1 гранями (объем вычислений $\approx N^2$),
- алгоритмы работающие в пространстве изображения, когда для каждого пиксела изображения определяется какая из N граней объекта видна (при разрешении экрана M×M объем вычислений $\approx M^2 \times N$).

Вычислительная сложность

$$M^2 \times N \approx N^2 \Longrightarrow M^2 \approx N \Longrightarrow 1000\ 000 \approx N$$

Детально представленые сцены — до первых сотен тысяч граней.

Алгоритм Роберта

Удаление невидимых линий в пространстве объектов.

Алгоритм удаления поверхностей с Z-буфером

Работает в пространстве изображения для тел, ограниченных неупорядоченными плоскими многоугольниками.

∀ пиксела хранится Z-координата (глубина)

1 УДАЛЕНИЕ СКРЫТЫХ ЛИНИЙ И ПОВЕРХНОСТЕЙ

1.1 Классификация методов удаления невидимых частей

Методы удаления невидимых частей сцены можно классифицировать:

1. По выбору удаляемых частей:
   удаление невидимых линий, ребер, поверхностей, объемов.
2. По порядку обработки элементов сцены:
   удаление в произвольном порядке и в порядке, определяемом процессом визуализации.
3. По системе координат:
   - алгоритмы работающие в пространстве объектов, когда каждая из N граней объекта сравнивается с остальными N-1 гранями (объем вычислений растет как N^2),
   - алгоритмы работающие в пространстве изображения, когда для каждого пиксела изображения определяется какая из N граней объекта видна (при разрешении экрана M×M объем вычислений растет как $M^2 \times N$).

1.2 Алгоритмы удаления линий

Применение — представление каркасных моделей. При этом не используется основное ценное качество растрового дисплея — возможность закраски поверхностей. В этой связи основная область применения — векторные устройства, но могут применяться и в растровых для ускорения процесса визуализации.

Наиболее известный ранний алгоритм — алгоритм Робертса (1963 г.). Работает с только выпуклыми телами в пространстве объектов. Каждый объект сцены представляется многогранным телом, полученным в результате пересечения плоскостей. Т.е. тело описывается списком граней, состоящих из ребер, которые в свою очередь образованы вершинами.

Вначале из описания каждого тела удаляются нелицевые плоскости, экранированные самим телом. Затем каждое из ребер сравнивается с каждым телом для определения видимости или невидимости. Т.е. объем вычислений растет как квадрат числа объектов в сцене. Наконец вычисляются новые ребра, полученные при протыкании телами друг друга.

1.3 Алгоритм удаления поверхностей с Z-буфером

Алгоритм предложен Эдом Кэтмулом и представляет собой обобщение буфера кадра. Обычный буфер кадра хранит коды цвета для каждого пиксела в пространстве изображения. Идея алгоритма состоит в том, чтобы для каждого пиксела дополнительно хранить еще и координату Z или глубину. При занесении очередного пиксела в буфер кадра значение его Z-координаты сравнивается с Z-координатой пиксела, который уже находится в буфере. Если Z-координата нового пиксела больше, чем координата старого, т.е. он ближе к наблюдателю, то атрибуты нового пиксела и его Z-координата заносятся в буфер, если нет, то ни чего не делается.

Этот алгоритм наиболее простой из всех алгоритмов удаления невидимых поверхностей, но требует большого объема памяти. Данные о глубине для реалистично-

сти изображения обычно достаточно иметь с разрядностью порядка 20 бит. В этом случае при изображении нормального телевизионного размера в 768×576 пикселов для хранения Z-координат необходим объем памяти порядка 1 Мбайта. Суммарный объем памяти при 3 байтах для значений RGB составит более 2.3 Мбайта.

Время работы алгоритма не зависит от сложности сцены. Многоугольники, составляющие сцену, могут обрабатываться в произвольном порядке. Для сокращения затрат времени нелицевые многоугольники могут быть удалены. По сути дела алгоритм с Z-буфером — некоторая модификация уже рассмотренного алгоритма заливки многоугольника. Если используется построчный алгоритм заливки, то легко сделать пошаговое вычисление Z-координаты очередного пиксела, дополнительно храня Z-координаты его вершин и вычисляя приращение dz Z-координаты при перемещении вдоль X на dx, равное 1. Если известно уравнение плоскости, в которой лежит обрабатываемый многоугольник, то можно обойтись без хранения Z-координат вершин. Пусть уравнение плоскости имеет вид:

$$A \cdot x + B \cdot y + C \cdot z + D = 0.$$

Тогда при C не равном нулю

$$z = -(A \cdot x + B \cdot y + D)/C$$

Найдем приращение Z-координаты пиксела при шаге по X на dx, помня, что Y очередной обрабатываемой строки — константа.

$$dz = -(A \cdot (x + dx) + D)/C + (A \cdot x + D)/C = -A \cdot dx/C$$

но $dx = 1$, поэтому

$$dz = -A/C.$$

Основной недостаток алгоритма с Z-буфером — дополнительные затраты памяти. Для их уменьшения можно разбивать изображение на несколько прямоугольников или полос. В пределе можно использовать Z-буфер в виде одной строки. Понятно, что это приведет к увеличению времени, так как каждый прямоугольник будет обрабатываться столько раз, на сколько областей разбито пространство изображения. Уменьшение затрат времени в этом случае может быть обеспечено предварительной сортировкой многоугольников на плоскости.

Другие недостатки алгоритма с Z-буфером заключаются в том, что так как пикселы в буфер заносятся в произвольном порядке, то возникают трудности с реализацией эффектов прозрачности или просвечивания и устранением лестничного эффекта с использованием предфильтрации, когда каждый пиксел экрана трактуется как точка конечного размера и его атрибуты устанавливаются в зависимости от того какая часть пиксела изображения попадает в пиксел экрана. Но другой подход к устранению лестничного эффекта, основанный на постфильтрации — усреднении значений пиксела с использованием изображения с большим разрешением реализуется сравнительно просто за счет увеличения расхода памяти (и времени). В этом случае использу-

ются два метода. Первый состоит в том, что увеличивается разрешение только кадрового буфера, хранящего атрибуты пикселов, а разрешение Z-буфера делается совпадающим с размерами пространства изображения. Глубина изображения вычисляется только для центра группы усредняемых пикселов. Это метод неприменим, когда расстояние до наблюдателя имитируется изменением интенсивности пикселов. Во втором методе и кадровый и Z буфера имеют увеличенное разрешение и усредняются атрибуты пиксела, так и его глубина.

Общая схема алгоритма с Z-буфером:

Инициализировать кадровый и Z-буфера. Кадровый буфер закрашивается фоном. Z-буфер закрашивается минимальным значением Z.

Выполнить преобразование каждого многоугольника сцены в растровую форму. При этом для каждого пиксела вычисляется его глубина z. Если вычисленная глубина больше, чем глубина, уже имеющаяся в Z-буфере, то занести в буфера атрибуты пиксела и его глубину, иначе никаких занесений не выполнять.

Выполнить, если это было предусмотрено, усреднение изображения с понижением разрешения.

1.4 Построчный алгоритм с Z-буфером

Рассмотрим теперь алгоритм с Z-буфером размером в одну строку, который представляет собой обобщение алгоритма построчной заливки многоугольника, представленный в процедурах V_FP0 и V_FP1 в приложениях. Модификация должна учесть то, что для каждой строки сканирования теперь может обрабатываться не один многоугольник.

Общая схема такого алгоритма следующая:

1. Подготовка данных.

   Для каждого многоугольника определить максимальную Y-координату.

   Занести многоугольник в группу многоугольников, соответствующую данной Y-координате.
2. Собственно заливка.

1.5 Алгоритм разбиения области Варнока

Алгоритм работает в пространстве изображения и анализирует область на экране дисплея (окно) на наличие в них видимых элементов. Если в окне нет изображения, то оно просто закрашивается фоном. Если же в окне имеется элемент, то проверяется достаточно ли он прост для визуализации. Если объект сложный, то окно разбивается на более мелкие, для каждого из которых выполняется тест на отсутствие и/или простоту изображения. Рекурсивный процесс разбиения может продолжаться до тех пор пока не будет достигнут предел разрешения экрана.

Можно выделить 4 случая взаимного расположения окна и многоугольника (рис. ??):
многоугольник целиком вне окна,
многоугольник целиком внутри окна,
многоугольник пересекает окно,
многоугольник охватывает окно.

В четырех случаях можно сразу принять решение о правилах закраски области экрана:

все многоугольники сцены — внешние по отношению к окну. В этом случае окно закрашивается фоном;

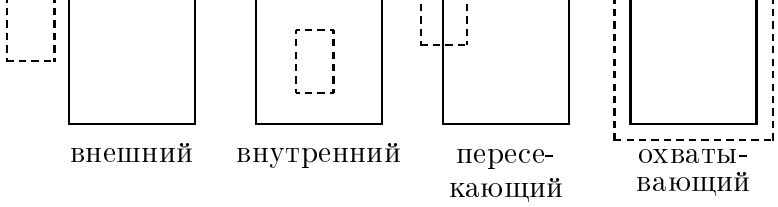


Рис. 1: Соотношения между окном экрана (сплошная рамка) и многоугольником (штриховая рамка)

имеется всего один внутренний или пересекающий многоугольник. В этом случае все окно закрашивается фоном и затем часть окна, соответствующая внутреннему или пересекающему окну закрашивается цветом многоугольника;

имеется единственный охватывающий многоугольник. В этом случае окно закрашивается его цветом.

имеется несколько различных многоугольников и хотя бы один из них охватывающий. Если при этом охватывающий многоугольник расположен ближе остальных к наблюдателю, то окно закрашивается его цветом.

В любых других случаях процесс разбиения окна продолжается. Легко видеть, что при растре 1024×1024 и делении стороны окна пополам требуется не более 10 разбиений. Если достигнуто максимальное разбиение, но не обнаружено ни одного из приведенных выше четырех случаев, то для точки с центром в полученном минимальном окне (размером в пиксел) вычисляются глубины оставшихся многоугольников и закраску определяет многоугольник, наиболее близкий к наблюдателю. При этом для устранения лестничного эффекта можно выполнить дополнительные разбиения и закрасить пиксел с учетом всех многоугольников, видимых в минимальном окне.

Первые три случая идентифицируются легко. Последний же случай фактически сводится к поиску охватывающего многоугольника, перекрывающего все остальные многоугольники, связанные с окном. Проверка на такой многоугольник может быть выполнена следующим образом: в угловых точках окна вычисляются Z-координаты для всех многоугольников, связанных с окном. Если все четыре такие Z-координаты охватывающего многоугольника ближе к наблюдателю, чем все остальные, то окно закрашивается цветом соответствующего охватывающего многоугольника. Если же нет, то мы имеем сложный случай и разбиение следует продолжить.

Очевидно, что после разбиения окна охватывающие и внешние многоугольники наследуются от исходного окна. Поэтому необходимо проверять лишь внутренние и пересекающие многоугольники.

Из изложенного ясно, что важной частью алгоритма является определение расположения многоугольника относительно окна.

Проверка на то что многоугольник внешний или внутренний относительно окна для случая прямоугольных окон легко реализуется использованием прямоугольной оболочки многоугольника и сравнением координат. Для внутреннего многоугольника должны одновременно выполняться условия:

$$\mathrm{Xmin} \geq \mathrm{W} \quad \mathrm{Xmax} \leq \mathrm{W} \quad \mathrm{Ymin} \geq \mathrm{W} \quad \mathrm{Ymax} \leq \mathrm{W},$$

здесь Xmin,Xmax,Ymin,Ymax — ребра оболочки
Wл, Wп, Wн, Wв — ребра окна

Для внешнего многоугольника достаточно выполнение любого из следующих условий:

$$\mathbf{Xmin > X,\ \ Xmax < W,\ \ Ymin > W,\ \ Ymax < W}$$

Таким способом внешний многоугольник, охватывающий угол окна не будет идентифицирован как внешний (см. рис. ??).

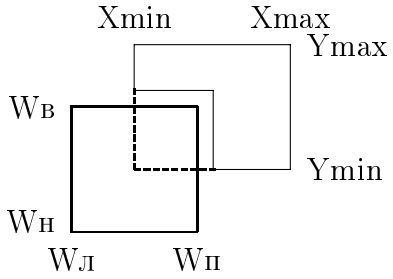


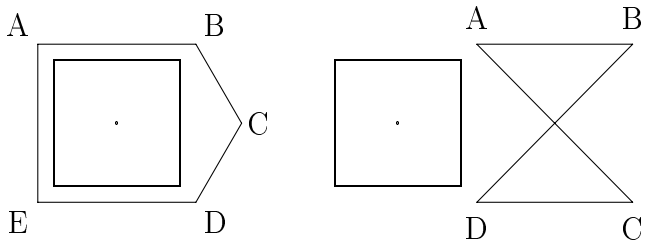
Рис. 2: Ошибочное определение внешнего многоугольника как пересекающего при использовании прямоугольной оболочки

Проверка на пересечение окна многоугольником может быть выполнена проверкой на расположение всех вершин окна по одну сторону от прямой, на которой расположено ребро многоугольника. Пусть ребро многоугольника задано точками P1(x1,y1,z1) и P2(x2,y2,z2), а очередная вершина окна задается точкой P3(x3,y3,z3). Векторное произведение вектора P1P3 на вектор P1P2, равное (x3-x1)(y2-y1) — (y3-y1)(x2-x1) будет меньше 0, равно 0 или больше 0, если вершина лежит слева, на или справа от прямой P1P2. Если знаки различны, то окно и многоугольник пересекаются. Если же все знаки одинаковы, то окно лежит по одну сторону от ребра, т.е. многоугольник может быть либо внешним, либо охватывающим.

Вернемся к примеру ??. Такой многоугольник рассмотренными тестами не был идентифицирован ни как

внутренний ни как пересекающий. Т.е. он может быть либо внешним, либо охватывающим. Для завершающей классификации может использоваться тест с подсчетом угла, рассматривавшийся ранее для определения нахождения точки внутри/вне многоугольника. В этом тесте вычисляется суммарный угол, на который повернется луч, исходящий из некоторой точки окна (обычно центра), при последовательном обходе вершин многоугольника.

Если суммарный угол равен 0, то многоугольник — внешний. Если же угол равен N×360°, то многоугольник охватывает окно N раз. Простейшая иллюстрация этого теста приведена на рис. ??.



а) многоугольник ABCDE — охватывающий Суммарный угол равен 360

б) многоугольник ABCD — внешний Суммарный угол равен 0

Рис. 3: Тест на охватывающий/внешний многоугольник

1.6 Построчный алгоритм Уоткинса

В алгоритмах построчного сканирования результирующее изображение генерируется построчно причем, подобно ранее рассмотренному алгоритму построчной заливки многоугольника, используется связность соседних растровых строк изображения. Отличие состоит в том, что учитываются все, а не один многоугольник.

Алгоритм работает в пространстве изображения с окном высотой в одну строку и шириной в экран, тем самым трехмерная задача сводится к двумерной.

Последовательность шагов алгоритма:
построение списка ребер,
построение списка многоугольников,

построение списка активных ребер — создается таблица ребер, включающая все негоризонтальные ребра многоугольников, причем элементы таблицы по значению Y-координаты отсортированы по группам.

1.7 Алгоритм трассировки лучей

При рассмотрении этого алгоритма предполагается, что наблюдатель находится на положительной полуоси Z, а экран дисплея перпендикулярен оси Z и располагается между объектом и наблюдателем.

Удаление невидимых (скрытых) поверхностей в алгоритме трассировки лучей выполняется следующим образом:
сцена преобразуется в пространство изображения,

из точки наблюдения в каждый пиксел экрана проводится луч и определяется какие именно объекты сцены пересекаются с лучом,

вычисляются и упорядочиваются по Z координаты точек пересечения объектов с лучом. В простейшем случае для непрозрачных поверхностей без отражений и преломлений видимой точкой будет точка с максимальным значением Z-координаты. Для более сложных случаев требуется сортировка точек пересечения вдоль луча.

Ясно, что наиболее важная часть алгоритма — про-

цедура определения пересечения, которая в принципе выполняется Rx × Ry × N раз (здесь Rx,Ry — разрешение дисплея по X и Y, соответственно, а N — количество многоугольников в сцене).

Очевидно, что повышение эффективности может достигаться сокращением времени вычисления пересечений и избавлением от ненужных вычислений. Последнее обеспечивается использованием геометрически простой оболочки, объемлющей объект — если луч не пересекает оболочку, то не нужно вычислять пересечения с ним многоугольников, составляющих исследуемый объект.

При использовании прямоугольной оболочки определяется преобразование, совмещающее луч с осью Z. Оболочка подвергается этому преобразованию, а затем попарно сравниваются знаки Xmin с Xmax и Ymin с Ymax. Если они различны, то есть пересечение луча с оболочкой (см. рис. ??)

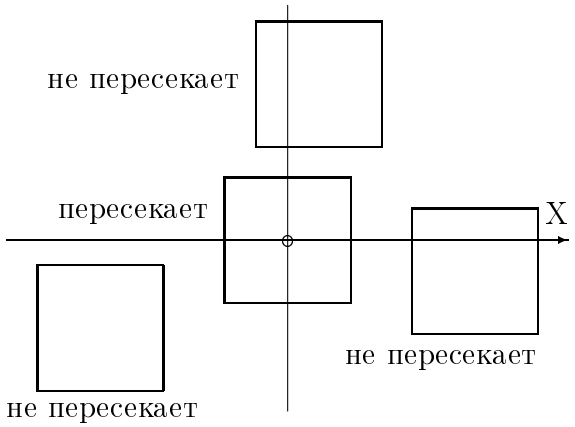


Рис. 4: Определение пересечения луча и оболочки

При использовании сферической оболочки для определения пересечения луча со сферой достаточно сосчитать расстояние от луча до центра сферы. Если оно больше радиуса, то пересечения нет. Параметрическое

уравнение луча, проходящего через две точки P1(x1,y1,z1) и P2(x2,y2,z2), имеет вид:

$$P(t) = P1 + (P2 - P1) \times t$$

Минимальное расстояние от точки центра сферы P0(x0,y0,z0) до луча равно:

$$d^2 = (x - x0)^2 + (y - y0)^2 + (z - z0)^2$$

Этому соответствует значение t:

$$t = -\frac{(x2 - x1) \cdot (x1 - x0) + (y2 - y1) \cdot (y1 - y0) + (z2 - z1) \cdot (z$$
$$(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2$$

Если $d2 > R2$, то луч не пересекает объекты, заключенные в оболочку.

Дальнейшее сокращение расчетов пересечений основывается на использовании групп пространственно связанных объектов. Каждая такая группа окружается общей оболочкой. Получается иерархическая последовательность оболочек, вложенная в общую оболочку для всей сцены. Если луч не пересекает какую-либо оболочку, то из рассмотрения исключаются все оболочки, вложенные в нее и, следовательно, объекты. Если же луч пересекает некоторую оболочку, то рекурсивно анализируются все оболочки вложенные в нее.

Наряду с вложенными оболочками для сокращения расчетов пересечений используется отложенное вычисление пересечений с объектами. Если обнаруживается, что объект пересекается лучом, то он заносится в специальный список пересеченных. После завершения обработки всех объектов сцены объекты, попавшие в список

пересеченных упорядочиваются по глубине. Заведомо невидимые отбрасываются а для оставшихся выполняется расчет пересечений и отображается точка пересечения наиболее близкая к наблюдателю.

Дополнительное сокращение объема вычислений может достигаться отбрасыванием нелицевых граней, учетов связности строк растрового разложения и т.д.

Для сокращения времени вычислений собственно пересечений предложено достаточно много алгоритмов, упрощающих вычисления для определенной формы задания поверхностей.

Реалистичное представление сцен

- Модели освещения
- Механизмы отражения света
- Модели закраски
- Прозрачность
- Тени
- Фактура
- Трассировка лучей
- Излучательность

Основные направления

- синтез реалистичных изображений,
- реалистическое оживление синтезированных объектов.

Источники света

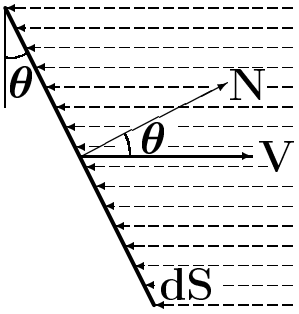
- излучающие и отражающие источники
- точечные источники
- распределенные источники
- рассеянный свет

Поверхности

- отражающие
- поглощающие
- полупрозрачные
- рассеивающие

Диффузное отражение

Закон Ламберта — падающий свет рассеивается во все стороны с одинаковой интенсивностью. Освещенность точки пропорциональна доле ее площади, видимой от источника.



$$\mathbf{Ir} = \mathbf{Ip}\cdot\mathbf{Pd}\cdot\cos(\theta),$$

Ir — интенсивность отраженного света,
Ip — интенсивность точечного источника,
$0 \le \mathbf{Pd} \le 1$ — коэффициент диффузного отражения, зависящий от материала поверхности и длины волны,
$0 \le \theta \le \pi/2$ — угол между направлением света и нормалью к поверхности.

Учет рассеянного света

$$\mathbf{I} = \mathbf{Ir}\cdot\mathbf{Pr} + \mathbf{Ip}\cdot\mathbf{Pd}\cdot\cos(\theta),$$

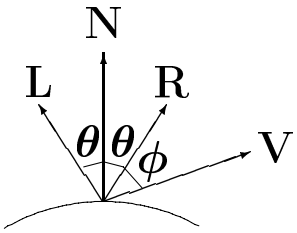
Ir — интенсивность рассеянного света,
$0 \le \mathbf{Pr} \le 1$ — коэффициент отражения рассеянного света.

Учет расстояния

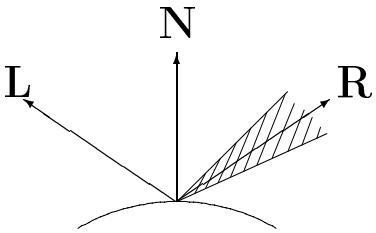
$$\mathbf{I = Ir \cdot Pr + \frac{Ip \cdot Pd \cdot \cos(\theta)}{d + K}},$$

d — расстояние от центра проекции до объекта,
при параллельной проекции d — расстояние от объекта, ближайшего к наблюдателю,
K — произвольная константа.

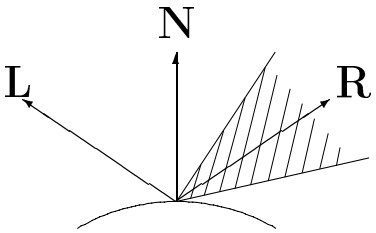
Зеркальное отражение



а)
Зеркальное отражение



б)
Отражение от блестящей поверхности



в)
Отражение от тусклой поверхности

$\vec{\mathbf{L}}$ — ед. вектор направления на источник света.
$\vec{\mathbf{N}}$ — нормаль к поверхности.
$\vec{\mathbf{R}}$ — ед. вектор направления идеального отражения.
$\vec{\mathbf{V}}$ — ед. вектор направления к наблюдателю.

Эмпирическая модель Фонга:

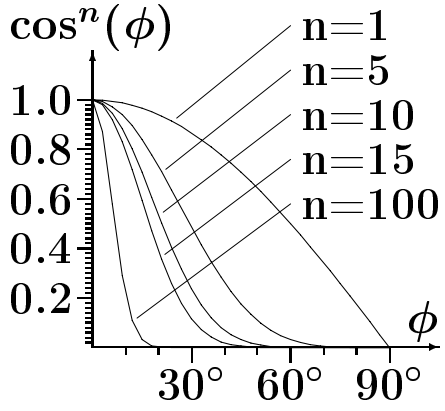
$$\mathbf{Is = Ip \cdot W(\lambda, \theta) \cdot \cos^n(\phi)},$$

$\mathbf{W(\lambda, \theta)}$ — кривая отражения,
$\mathbf{-\pi/2 \le \phi \le \pi/2}$,
$\mathbf{1 \le n \le 200}$,

Для идеального отражателя $\mathbf{n} = \infty$.

Для тусклых, негладких поверхностей типа мела или сажи $\mathbf{n} \approx 1$.

Зависимость $\cos^n(\phi)$ от значения параметра отражения n:



Часто $\mathbf{W}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ заменяется константой Ks, такой чтобы полученная картина была субъективно приемлема.

Суммарная модель освещения:

$$\mathbf{I} = \mathbf{Ir} \cdot \mathbf{Pr} + \frac{\mathbf{Ip}}{\mathbf{d} + \mathbf{K}}(\mathbf{Pd} \cdot \mathbf{cos}(\boldsymbol{\theta}) + \mathbf{W}(\boldsymbol{\lambda}, \boldsymbol{\theta}) \cdot \mathbf{cos^n}(\boldsymbol{\phi})).$$

Или при замене $\mathbf{W}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ на константу Ks:

$$\mathbf{I} = \mathbf{Ir} \cdot \mathbf{Pr} + \frac{\mathbf{Ip}}{\mathbf{d} + \mathbf{K}}(\mathbf{Pd} \cdot \mathbf{cos}(\boldsymbol{\theta}) + \mathbf{Ks} \cdot \mathbf{cos^n}(\boldsymbol{\phi})).$$

Или:

$$\mathbf{I} = \mathbf{Ir} \cdot \mathbf{Pr} + \frac{\mathbf{Ip}}{\mathbf{d} + \mathbf{K}}(\mathbf{Pd} \cdot \vec{\mathbf{L}} \cdot \vec{\mathbf{N}} + \mathbf{Ks} \cdot (\vec{\mathbf{R}} \cdot \vec{\mathbf{V}})^{\mathbf{n}}).$$

$\vec{\mathbf{L}}$, $\vec{\mathbf{N}}$, $\vec{\mathbf{R}}$ и $\vec{\mathbf{V}}$ — нормированные векторы направлений падения, нормали, отражения, и наблюдения.

Если источник света находится на бесконечности, то для данного плоского многоугольника $\vec{L} \cdot \vec{N}$ равно константе, а $\vec{R} \cdot \vec{V}$ меняется в пределах многоугольника.

Для поверхностей, представленных например в виде бикубических кусков, каждое произведение меняется в пределах куска.

Фонг предложил алгоритм пошагового вычисления по рассмотренной модели, существенно снижающий затраты.

Модели с микрогранями

1. Отражающая поверхность представлена в виде плоских микрограней.
2. Ориентации нормалей к граням относительно нормали к средней линии поверхности задаются некоторым распределением, например, Гаусса.

Модели закраски

- однотонная (и источник и наблюдатель на бесконечности)
- метод Гуро
- метод Фонга

Прозрачность

- без учета преломления
- с учетом преломления

Суммарная закраска:

$$I = k \cdot I + (1 - k) \cdot I,$$

$0 \leq k \leq 1$ — характеризует прозрачность ближнего многоугольника.
Если $k = 1$, то он непрозрачен.
Если $k = 0$, то ближний многоугольник полностью прозрачен.
Iб — интенсивность для пиксела ближнего многоугольника.
Iд — дальнего.

Тени

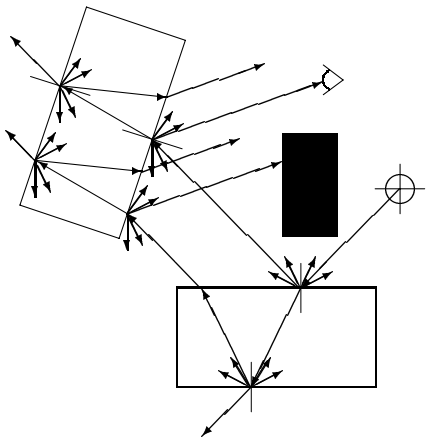
Объект невидимый из источника света находится в тени.

Шаги алгоритма:

1. Определяются все многоугольники, видимые из точки освещения.
2. Удаление поверхностей невидимых из точки зрения.
3. Закраска многоугольников.
   Если видим из источника освещения, то учитываются диффузное и зеркальное отражения и рассеянный свет.
   Если невидим, то многоугольник в тени и надо учитывать рассеянное освещение.

Трассировка лучей

- прямая трассировка лучей,
- обратная трассировка лучей.



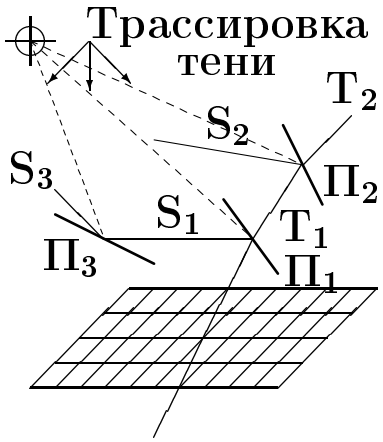
Прямая трассировка лучей

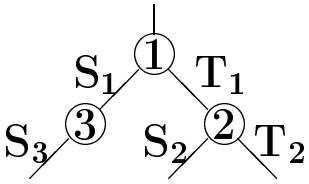
Расчет освещения сцены:

- от всех источ ников света испускаются лучи во всех направлениях;
- расчитываются преломление и отражение каждого луча, в том числе и отраженного, т.е. каждая точка сцены может освещаться либо напрямую источником, либо отраженным светом;
- часть лучей, попавшая в глаз наблюдателя, сформирует в нем изображение сцены.

Изображение формирует только малая часть лучей.

Обратная трассировка лучей





Расчет освещения сцены:

- отслеживаются лучи, проходящие из глаза наблюдателя через каждый пиксел экрана в сцену;
- на каждой поверхности сцены, на которую попадает луч, формируются отраженный и преломленный лучи;
- каждый такой луч рекурсивно отслеживается, чтобы определить пересекаемые поверхности;
- ветвление прекращается если:
  – луч вышел за пределы сцены;
  – луч пришел на источник света;
  – луч попал на непрозрачный диффузный рассеиватель;
  – исчерпана память;
- в результате $\forall$ пиксела строится дерево пересечений. Ветви такого дерева — распространение луча, а узлы — пересечения с поверхностями;
- закраска пиксела определяется прохождением по дереву и вычислением вклада каждой пересеченной поверхности в соответствии с используемыми моделями отражения.

Две вычислительно интенсивные процедуры:

1. Построение дерева пересечений:
   ∀ пиксела каждый луч должен быть проверяться на пересечение с каждой поверхностью.

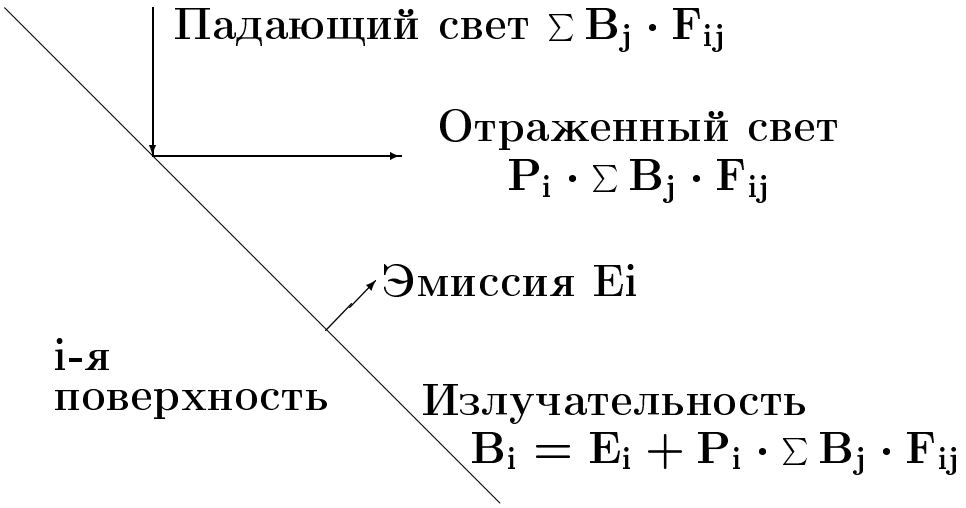
2. Тестирование тени:
   ∀ точки пересечения проверяется может ли она быть непосредственно освещеной от источников света.
   Для этого из точки проводится луч к источнику(ам) и проверяется есть ли пересечения с непрозрачными поверхностями.
   Если есть, то точка в тени.

Основные характеристики обратной трассировки

1. Высокий реализм.
2. Учет отражения, преломления, затухания.
3. Проблемы с диффузным отражением и преломлением. Так как диффузное отражение ведет к бесконечному числу лучей, то диффузное отражение и преломление или не учитывают или рассчитывают только для ближайшей к глазу поверхности.
4. Большой объем вычислений.
5. Интенсивность точки и глобальная освещенность вычисляются совместно, поэтому при смене точки наблюдений сцена д.б. перевычислена.
6. Затруднено моделирование распределенных источников света, т.к. расчет поточечный.
7. Используемые модели отражения — эмипирические и не обеспечивают сохранение энергии.

Излучательность

- Сцена — набор поверхностей, обменивающихся лучистой энергией.
- Поверхности идеально диффузны.
- Излучательность отдельной поверхности — самоизлучение и отраженный или пропущенный свет.



E_i — эмиссия = $\frac{-}{\cdot\cdot\cdot}$,

P_i — безразмерный коэффициент отражения,

B_i — полный уровень света, исходящего от поверхности,

F_{ij} — форм-фактор — часть энергии, исходящая от одной поверхности и достигающая другой.

Взаимодействие потоков света в сцене из n элементов постоянной излучательности:

$$\begin{bmatrix} 1-P_1 \cdot F_{11} & -P_1 \cdot F_{12} & \dots & -P_1 \cdot F_{1n} \\ 1-P_2 \cdot F_{21} & -P_2 \cdot F_{22} & \dots & -P_2 \cdot F_{2n} \\ & \dots & & \\ 1-P_n \cdot F_{n1} & -P_n \cdot F_{n2} & \dots & -P_n \cdot F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{bmatrix}.$$

Система уравнений решается итерационным методом.

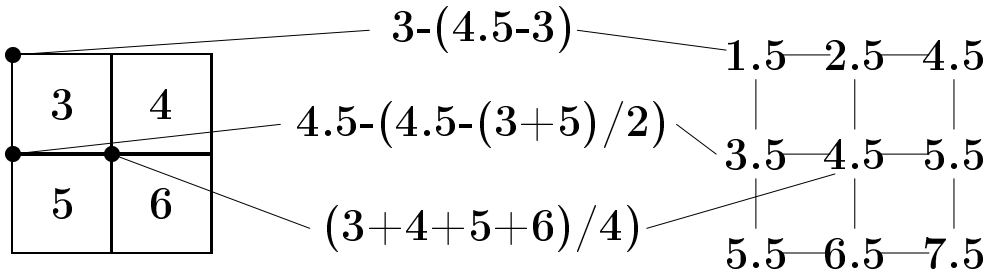
Начальное приближения для излучательности — эмиссия поверхности:

$B_i^0 = E_i$,

последующие приближения:

$B_i^{k+1} = E_i + P_i \sum_{j=1}^{n} F_{ij} B_j^k$.

Вычисление излучательностей вершин:

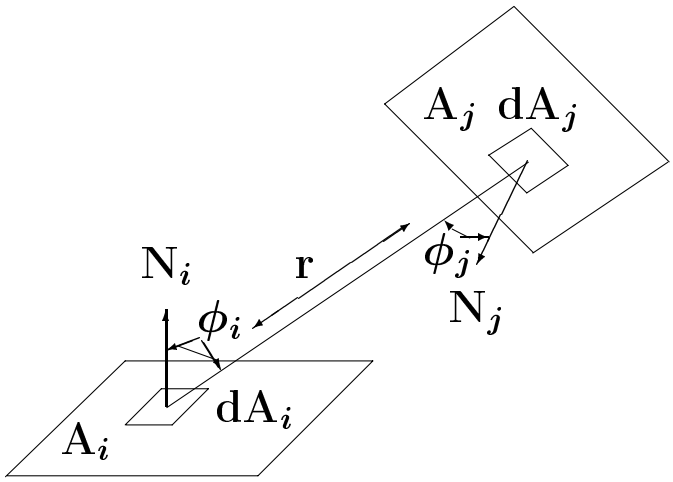


Вычисление форм факторов

Для двух неперекрытых элементарных площадок i и j форм-фактор определяет часть световой энергии, исходящей из одной площадки на другую, и имеет вид:

$$F_{dAi,dAj} = \frac{\cos(\phi_i) \cdot \cos(\phi_j)}{\pi \cdot r^2}$$

где r — расстояние между элементами dA_i и dA_j, ϕ_i и ϕ_j — углы между нормалями к элементам и соединяющим их отрезком



Форм-фактор конечного элемента площади A_j относительно элементарной площадки dA_i:

$$F_{dAi,Aj} = \int_{Aj} \frac{\cos(\phi_i) \cdot \cos(\phi_j)}{\pi \cdot r^2} \cdot dAj$$

Форм-фактор двух конечных неперекрытых площадок:

$$F_{ij} = F_{Ai,Aj} = \frac{1}{A_i} \int_{Ai} \int_{Aj} \frac{\cos(\phi_i) \cdot \cos(\phi_j)}{\pi \cdot r^2} \cdot dA_j \cdot dA_i$$

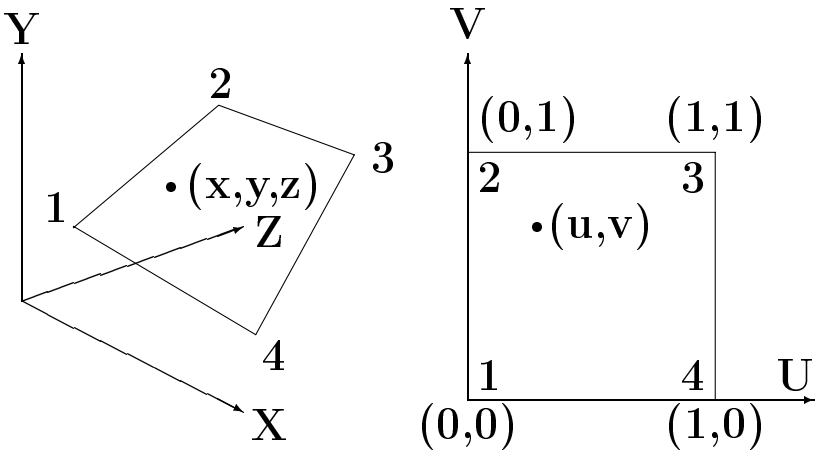
Учет перекрытия с помощью функции $0 \leq H_{ij} \leq 1$:

$$F_{ij} = F_{Ai,Aj} = \frac{1}{A_i} \int_{Ai} \int_{Aj} \frac{\cos(\phi_i) \cdot \cos(\phi_j)}{\pi \cdot r^2} \cdot H_{ij} \cdot dA_j \cdot dA_i$$

Метод не зависит от точки наблюдения.

Последовательность вывода:

- определяется какой фрагмент отображается в каждом пикселе экрана. При этом используются z-буфер и кадровый буфер.
- Отыскивается координата точки, отображаемой в текущий пиксел.
- Координата пересечения (x, y, z) преобразуется в координаты (u, v) на фрагменте.
- Значения излучательностей вершин используются при билинейной интерполяции излучательностей в пределах каждого фрагмента.



$$B(u,v) = (1-u)\cdot(1-v)\cdot B_1 + (1-u)\cdot B_2 + u\cdot v\cdot B_3 + u\cdot(1-v)\cdot B_4$$

Курс “Компьютерная графика”

1. **Цель курса**
2. **Организация занятий**
3. **Части курса**
   - **вводный курс;**
   - **основные алгоритмы;**
   - **архитектуры графических систем.**
4. **Результаты и контроль**
5. **Литература**

Архитектуры графических систем

6 Тем

1 тема, пройденная в предыдущем семестре, подчеркнута

1. ИНТЕРАКТИВНЫЕ СИСТЕМЫ МАШИННОЙ ГРАФИКИ
   - Графические языки высокого уровня
   - Синтаксические расширения алгоритмических языков
   - Процедурные графические языки
   - Языки диалога
2. АРХИТЕКТУРА ГРАФИЧЕСКИХ РАБОЧИХ СТАНЦИЙ
   - Компоненты современных растровых дисплейных систем
   - Видеопамять
   - Технические средства формирования изображений
   - RISC-процессор с графическим устройством (i860)
   - Высокоскоростные графические системы

3. СТАНДАРТИЗАЦИЯ В МАШИННОЙ ГРАФИКЕ

- **Международная деятельность по стандартизации в машинной графике**
- **Классификация и обзор международных стандартов**
- **Графические протоколы**

4. СИСТЕМЫ УПРАВЛЕНИЯ ПОЛЬЗОВАТЕЛЬСКИМ ИНТЕРФЕЙСОМ (UIMS)

- **Системы управления окнами (WMS)**
- **Инструментарий создания пользовательского интерфейса**
- **Системы управления интерфейсом пользователя**
- **Непосредственное манипулирование**
- **Пример реализации UIDS/UIMS**

5. VISC — ИНИЦИАТИВА

- **AVS — Прикладная система научной визуализации**
- **Архитектура системы прикладной визуализации**

6. ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ

ЛИТЕРАТУРА

1. Вельтмандер П.В. Введение в машинную графику: Учеб. пособие/ Новосиб. ун-т. Новосибирск, 1995. 77 с.
2. Вельтмандер П.В. Машинная графика. Вводный курс:Учеб. пособие в электронном виде.
3. Вельтмандер П.В. Машинная графика. Основные алгоритмы: Учеб. пособие в электронном виде.
4. Вельтмандер П.В. Машинная графика. Введение в архитектуры графических систем: Учеб. пособие в электронном виде.
5. Роджерс Д. Алгоритмические основы машинной графики. Пер. с англ. М.: Мир,1989. 512 с.
6. Павлидис Т. Алгоритмы машинной графики и обработки изображений. Пер. с англ. М.: Радио и связь, 1986.
7. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х книгах. Пер. с англ. М.: Мир, 1985.
8. Гилой В. Интерактивная машинная графика. Пер. с англ. М.: Мир, 1981.
9. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики. Пер. с англ. М.: Мир, 1976.
10. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. М.: “ДИАЛОГ–МИФИ”, 1995. 228 с.
11. Donald Hearn, M. Pauline Baker. Computer Graphics. Prentice Hall, 1994. 652 p.

Содержание:

- графические рабочие станции и суперстанции
- компоненты растровых дисплейных систем
- проектирование графических систем
- некоторые графические системы

Рабочие станции

Результат сбалансированного объединения технологий:

- построения процессоров,
- организации связи,
- организации ввода/вывода,
- работы с графическими объектами и устройствами.

Построены, в основном из стандартных компонент.
Операционная система, как правило, Unix.
Следуют концепции открытых систем.
Поддерживают 3D графику.
1980–90 — удвоение производительности за 2 года.

Суперстанции

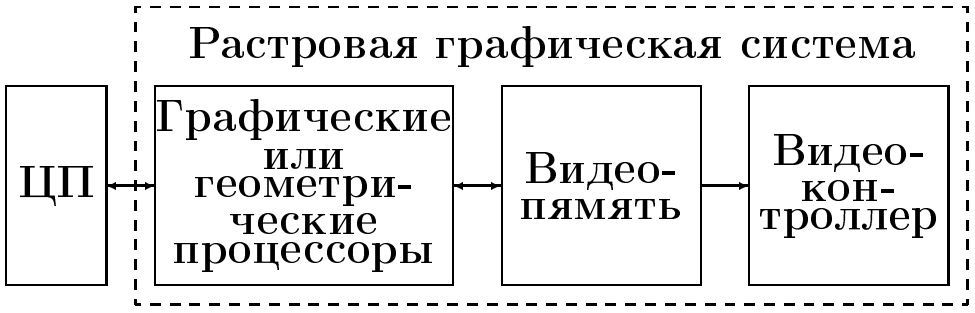
Объединение в одной системе возможностей рабочих станций (3D графика, интегрированность) и суперкомпьютеров (быстрый ввод/вывод, распараллеливание или векторизация вычислений).

Состав типичной суперстанции:

- один или несколько 32/64-битных ЦП;
- сопроцессоры с плавающей запятой и/или векторный;

- графическая подсистема с процессором, кадровым буфером и Z-буфером;
- не менее чем 32-битная внутренняя шина;
- сетевой контроллер (FDDI, Ethernet Token Ring);
- быстрый дисковый контроллер (IPI, SCSI ...);
- от 16 до 256 мегабайт внутренней памяти;
- стандартная шина ввода/вывода (VME, EISA, MCA ...) для подключения периферийных устройств;
- один или несколько асинхронных портов;
- монитор, клавиатура, мышь;
- Unix, X Window, NFS, PHIGS, GKS, C, Fortran, TCP-IP, эмуляторы графических терминалов, средства отладки ...

Компоненты растровых дисплейных систем



- графический процессор — генерация примитивов и BitBlt операции;
- видеопамять — хранение изображения;
- видеоконтроллер — отображение кадрового буфера, управление раскраской, некоторыми атрибутами и монитором.

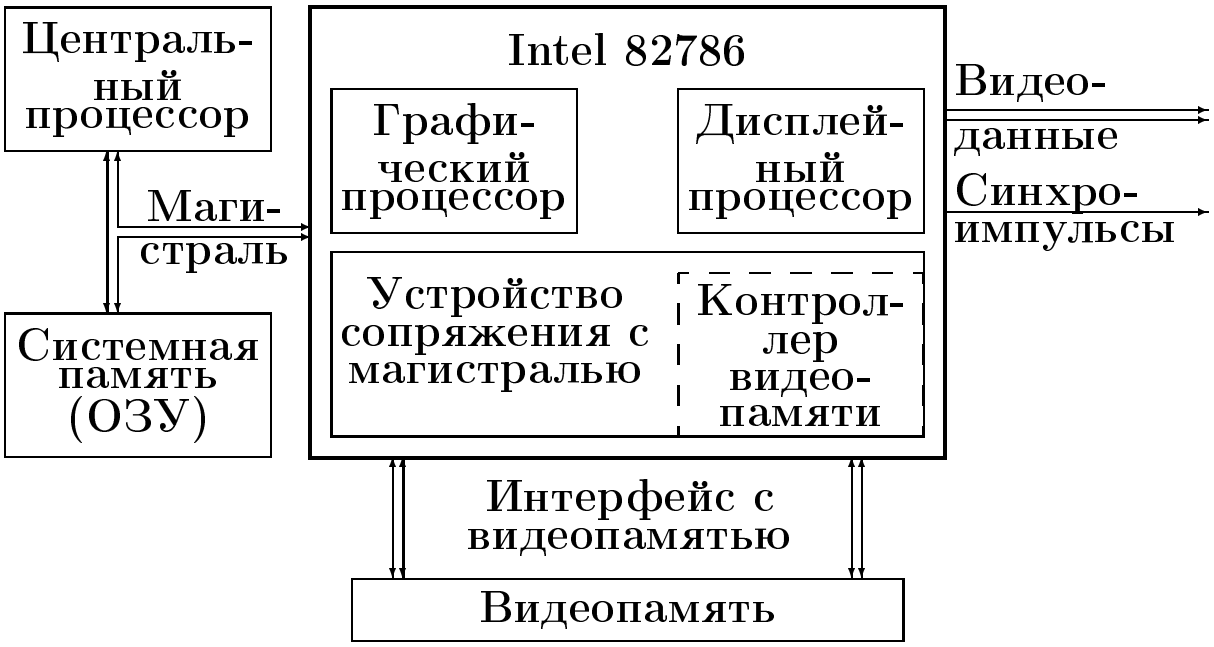
Подходы к проектированию графической системы:

максимум простоты, минимум гибкости — функционально завершенный графический сопроцессор, аппаратно реализующий основные графические примитивы.
Изменения алгоритмов сопроцессора невозможны.
Недостающие графические средства реализуются за счет доступа к видеопамяти из программы пользователя (Intel, i82786);

гибкость на этапе проектирования — построение графической системы из набора СБИС, реализующих строго регламентированные функции
(National Smiconductor, набор СБИС DP 85xx);

максимум гибкости — использование универсального процессора, программируемого на языке высокого уровня и аппаратно реализующего основные растровые операции
(Texas Instruments, TMS 34010, TMS 34020 и Intel, i860).

Графические системы на базе сопроцессора i82786



- неавтономная работа под управлением ЦП на базе микропроцессора Intel, который передает команды и данные, отрабатываемые сопроцессором.
- 22-разрядная шина адреса — до 4 М видеопамяти,
- 16-разрядная шина данных,
- системное тактирование — 10 МГц,
- видеотактирование — 25 МГц,
- тактирование дисплейного процессора:

МГц	бит/пиксел	МГц	бит/пиксел
25	8	100	2
50	4	200	1

- скорости генерации:

Тип	Мпикс./с	Тип	Мпикс./с
линии	2.5	окружности и дуги	2
BitBlt	30	заливка области	30

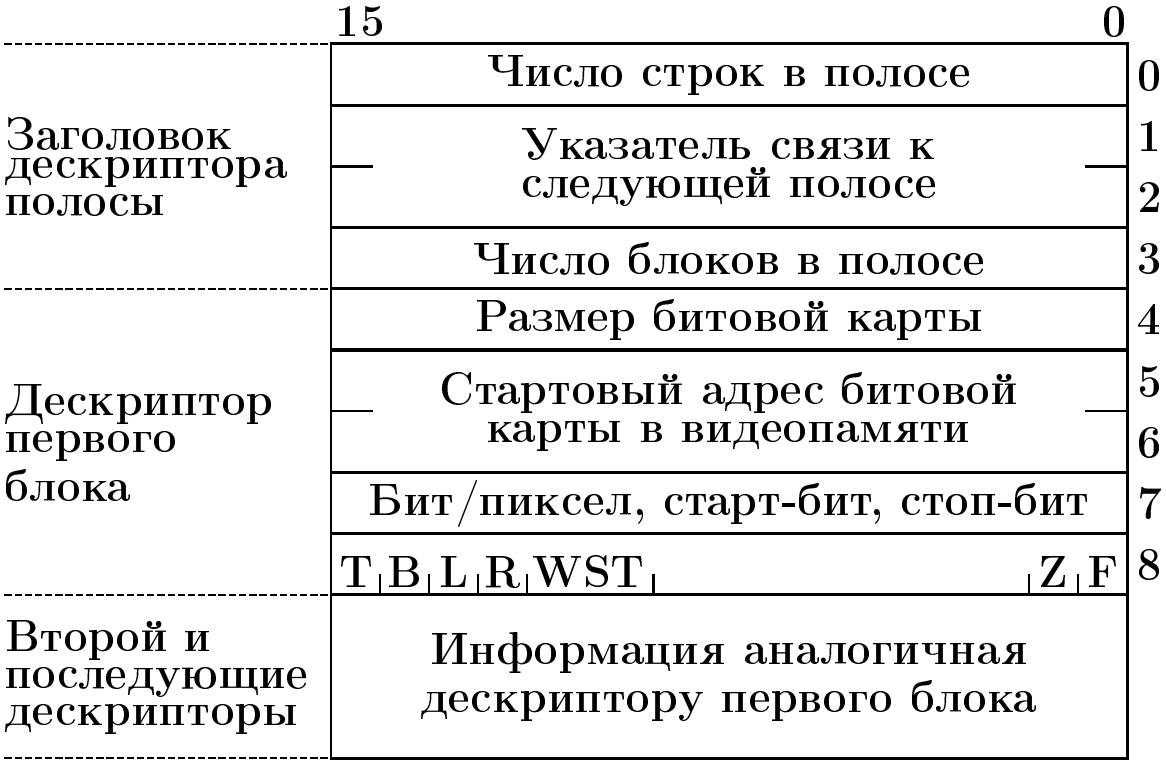
- построение символов — 25 млн. символов/с,
- разрешающая способность:
  от $640\times480\times8$ до $1024\times1024\times2$,
- программирование сигналов монитора — до 4096×4096

Набор команд:

- общее управление,
- управление курсором,
- управление атрибутами примитивов,
- генерация примитивов,
- блочная перепись и управление окнами.

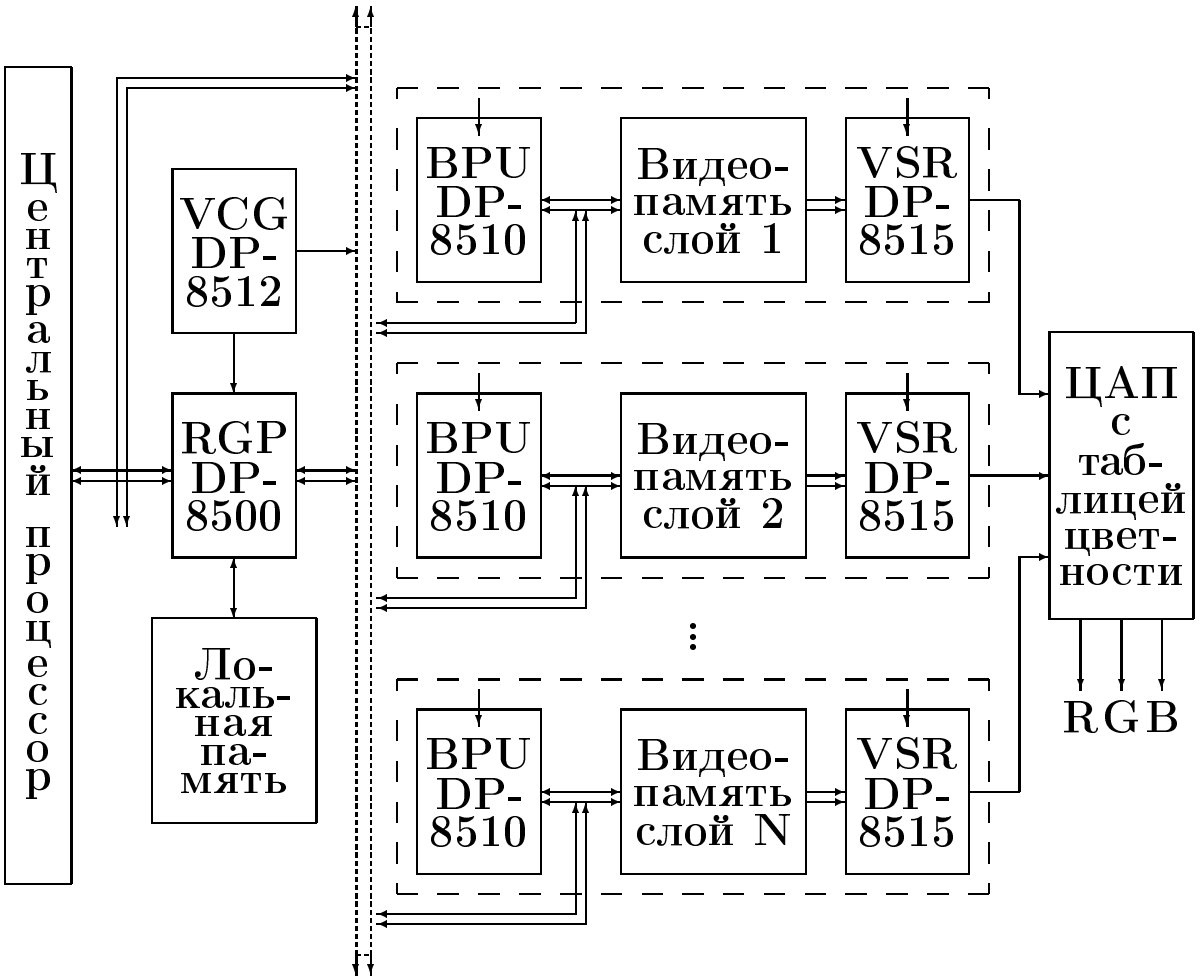
ЦП в видеопамяти формирует таблицу, описывающую конфигурацию экранного буфера.

Экранный буфер может быть разбит на полосы. Каждая строка может быть разбита на окна-сегменты. Минимальная ширина окна — 1/16 длины строки.

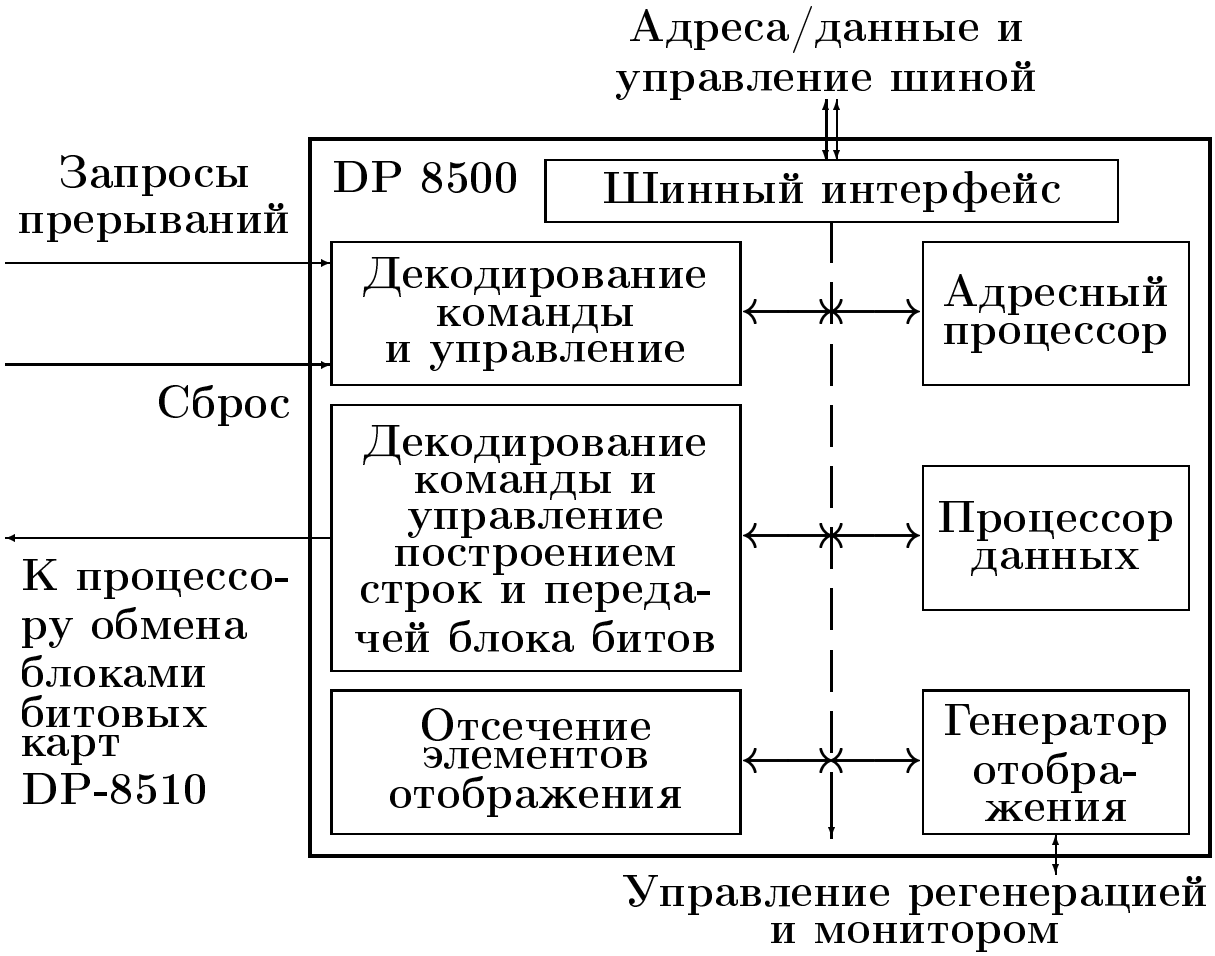


Разработчик сам определяет требуемые функции и реализует их, используя те или иные компоненты набора. Набор СБИС AGCS 85xx (Advanced Graphics Chip Set) фирмы National Semiconductor состоит из:

- DP-8500 — растровый графический процессор (RGP — Raster Graphics Processor),
- DP-8510 процессор обмена блоками информации (BPU — BitBlt Processing Unit),
- DP-8512 — генератор тактовых импульсов для вывода видеоизображения (VCG — Video Clock Generator),
- DP-8515 — высокоскоростной сдвиговый регистр (VSR — Video Shift Register).



Растровый графический процессор DP-8500



- тактирование 20 МГц,
- цикл шины 100 нс,
- производительность графической системы 10÷160 Мпикселов/с,
- адресное АЛУ, 28 разрядов, 16 регистров,
- АЛУ данных, 16 разрядов, 16 регистров,
- аппаратная генерация линий,
- аппаратное отсечение,
- аппаратное копирование блоков бит,
- видеопамять послойная, в глубину, смешанная,
- получение дисплейного файла от ЦП по флагу.

Расширение числа аппаратно реализованных функций мало приемлемо по следующим причинам:

1. Набор графических функций был бы жестко зафиксирован. Новые примитивы или старые, но с расширенными возможностями, не смогут быть поддержаны.

2. Аппаратная реализация означает “жесткий”выбор поддерживаемых атрибутов, (ТИП ЛИНИИ, ШИРИНА ЛИНИИ, ЦВЕТ ЛИНИИ, ПРОЗРАЧНОСТЬ и т.п.) т.е. некоторые редкие, но существенные для отдельных применений атрибуты будут опущены, например, ФОРМА КОНЦОВ ЛИНИИ (endpoint shape).

3. Высококачественная графика требует точного контроля над алгоритмами формирования изображений, например, для устранения ступенчатости. При аппаратной реализации потребуются дополнительные параметры. В программной реализации сглаживание выполняется при необходимости.

4. Формат дисплейного списка, или команд формирования изображений может варьироваться в соответствии с требованиями пользователя. (Например, отображение простых и/или высококачественных шрифтов.)

Единственный способ гибкого удовлетворения требований — программирование функций процессора, интерпретирующего графические команды.

Графический процессор TMS-34010

- внешнее 50 МГц тактирование, скорость — 6 MIPS;
- 120 инструкций, 4 типа данных — массивы упакованных пикселов, XY-координаты, прямоугольные окна и битовые поля;
- BitBlt инструкции с 16 логическими функциями;
- аппаратная реализация графических инструкций;
- логика маскирования и слияния цепочек пикселов;
- аппаратура определения левого единичного бита;
- компаратор окон в видеопамяти для поддержки отсечения;
- линейная и XY-адресация от 1 до 32 бит;
- число бит на пиксел — 4/8/16;
- видеопамять до 8 М.

Графический процессор TMS-34020

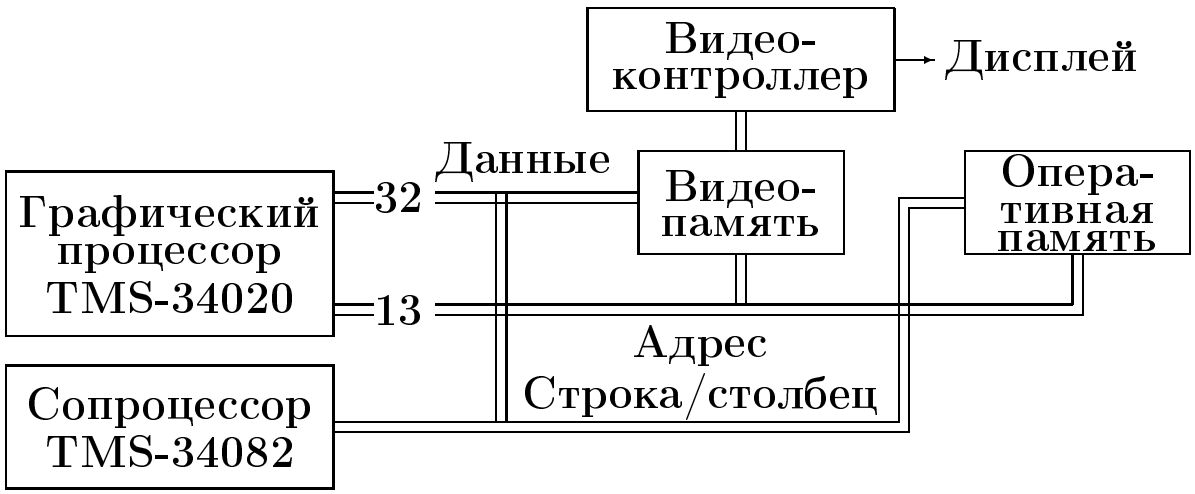
Преемник TMS-34010. Дополнительные возможности:

- высокоскоростное тактирование, скорость — 10 MIPS;
- дополнительные 3-х операндные инструкции PixBlt;

- BitBlt инструкции расширены 8 арифметическими функциями;
- скорость выполнения BitBlt операций — до 25 Мегапикселов/с;
- скорость рисования линий — до 5 Мегапикселов/с;
- (видео)память до 512 М.

Для использования в 3D-системах предусмотрен сопроцессор TMS-34082:

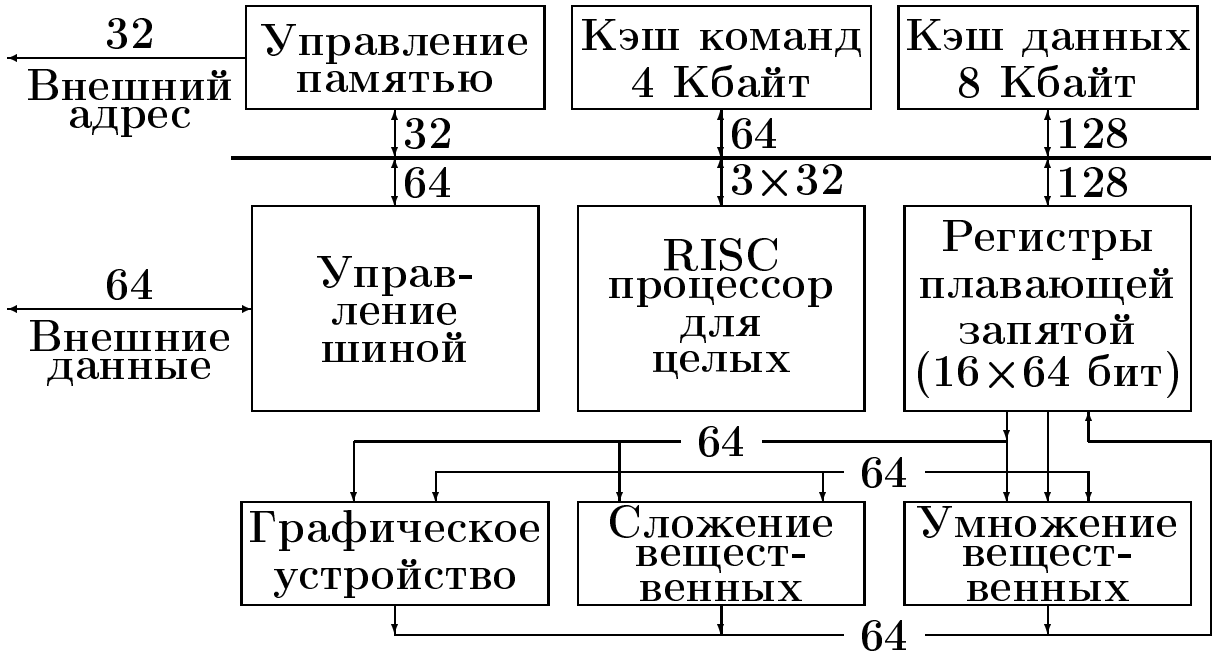
- скорость — 40 MFLOPS;
- обычные арифметические операции;
- 3D микропрограммные операции — умножение матриц 4×4, отсечение многоугольника, генерация 3D кубических сплайнов.



RISC процессор с графическим устройством i860

- тактирование 50 МГц;
- пиковое быстродействие — 40 MIPS и 80 MFLOPS;
- процессор для целых — управление системой и операции над 8, 16 и 32-х битовыми целыми;
- векторный и скалярный режимы для вещественных;

- графическое устройство обрабатывает до нескольких пикселов одновременно в 64-х битном слове;
- глубина пиксела — 8/16/32 бита;
- имеется режим поддержки 3D отображения;
- аппаратная поддержка сравнения в Z-буфере;
- закраска Гуро — 50000 треугольников/с;
- до 500 000 однородных преобразований/с.



Высокоскоростные графические системы

Кроме высокоскоростной генерации и манипулирования растровыми образами для формирования высокореалистичных картин в реальном времени требуются сбалансированные по времени:

- моделирование сцен,
- выполнение геометрических расчетов,
- расчет освещенностей.

Подходы для быстрого вычисления изображений:

- использование специализированной аппаратуры (Silicon Graphics);
- использование универсального вычислителя, дополненного средствами быстрого отображения (Stardent);

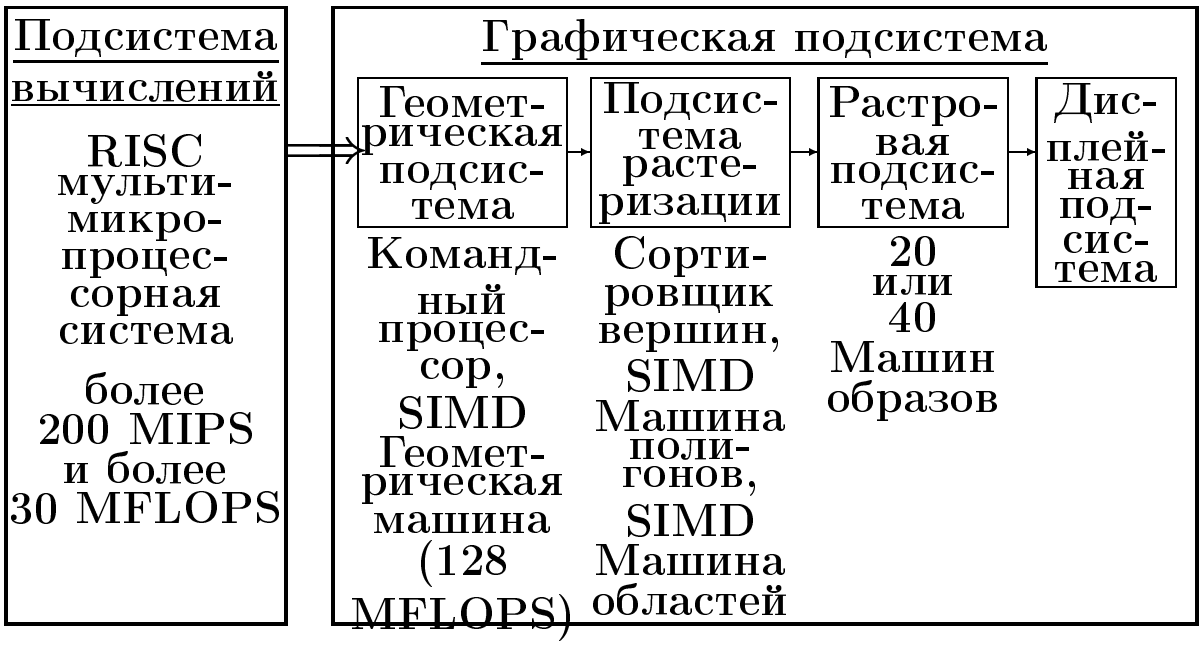
Рабочие (супер)станции Silicon Graphics

Джеймс Кларк (James Clark) к 1981 г разработал Геометрическую машину, ориентированную на моделирование 3D сцен.

1982 г. — основана фирма Silicon Graphics.

1984 г. — выход на рынок рабочих станций.

1988 г. — в рабочих станциях SG используются только RISC-процессоры фирмы MIPS.

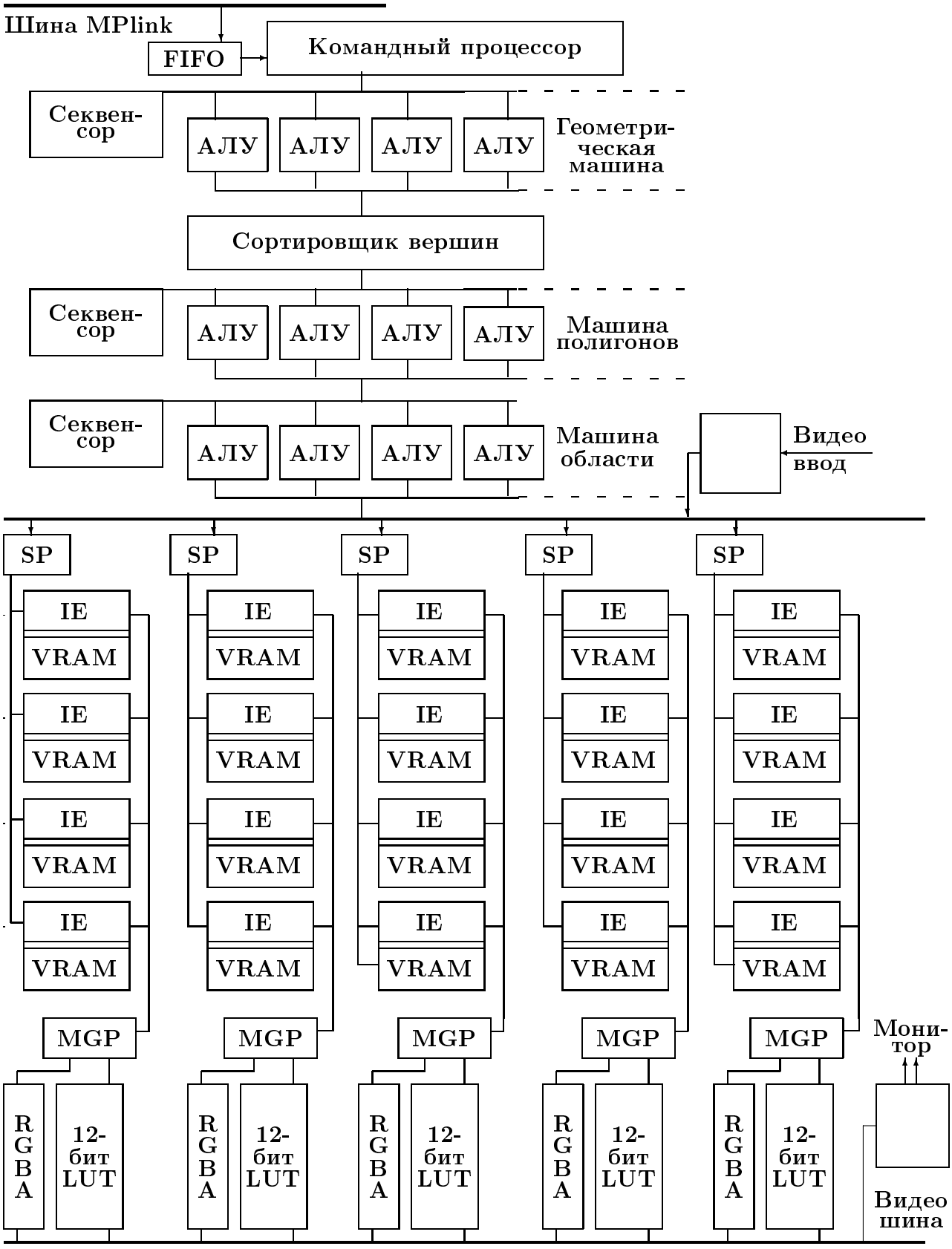


$> 10^6$ Z-буферизованных треугольников в сек

скорость доступа к кадровому буферу $> 20 \cdot 10^6$ пикс./с

Для каждого пиксела хранятся:

- R, G, B и Альфа-каналы по 8 бит,
- Z-координата,
- текстурные планы для R, G, B и Альфа-каналов,
- биты управления перекрытиями окон,
- биты задания способа отображения (полноцветное или индексное изображение, одно/двухкратная буферизация).

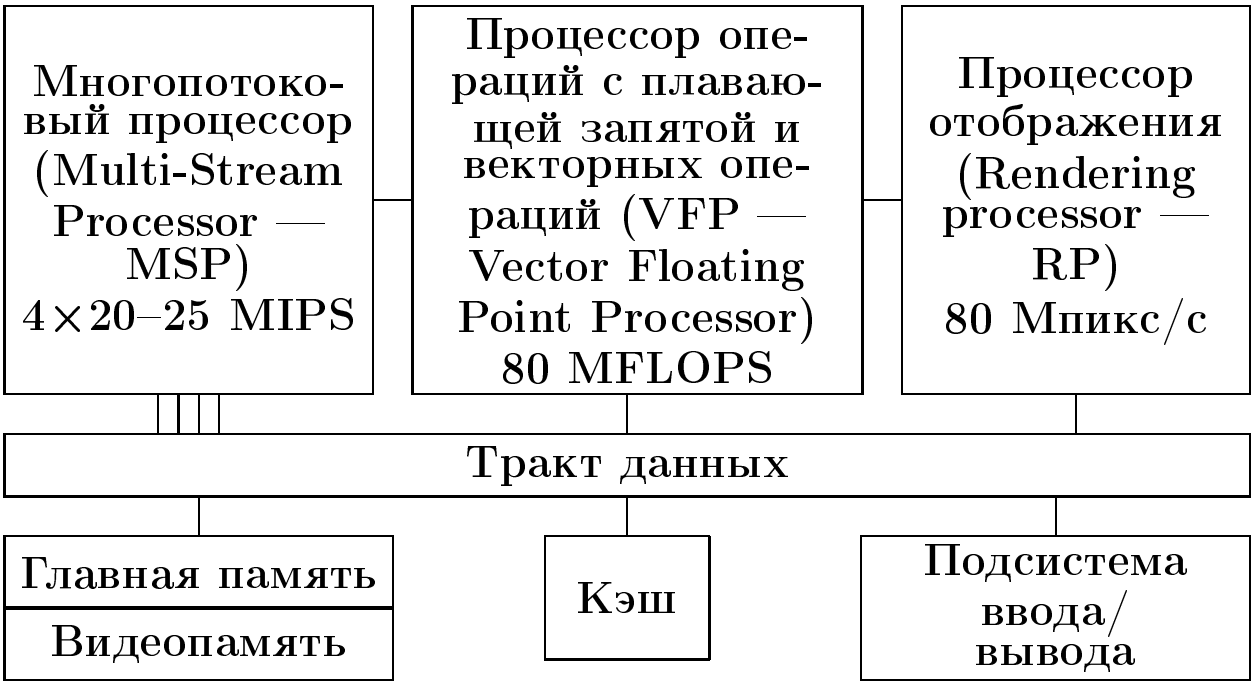

Шина MPlink
FIFO
Командный процессор
Секвен-
сор
АЛУ
АЛУ
АЛУ
АЛУ
Геометри-
ческая
машина
Сортировщик вершин
Секвен-
сор
АЛУ
АЛУ
АЛУ
АЛУ
Машина
полигонов
Секвен-
сор
АЛУ
АЛУ
АЛУ
АЛУ
Машина
области
Видео
ввод
SP
SP
SP
SP
SP
IE
VRAM
MGP
RGBA
12-
бит
LUT
Мони-
тор
Видео
шина

Рабочие (супер)станции с использованием универсального вычислителя

Многие вычисления при формировании изображений близки к обработке, требуемой в научных вычислениях (геометрические преобразования, расчеты освещенностей и т.д.).

Сочетание высокоскоростного вычислителя и средств быстрого отображения позволяет построить систему, пригодную и для вычислительно интенсивных заданий и для графики реального времени.

Такой подход был реализован в системе GS2000 фирмы Stardent.



VFP выполняет координатные преобразования (600 K 3D преобразований в сек), отсечение и вычисление освещенности.

RP ведет обработку изображений со скоростью 80 Мегапикселов/с. (600 K 3D клиппируемых векторов/с, длиной 10 пикселов и 160 K 100 пиксельных треугольников Гуро с Z-буферизацией).

- периоды развития графических средств и стандартизация
- Network Graphics Protocol
- Концептуальная модель графической системы
- Деятельность ISO, IEC
- Классификация стандартов
- Core System
- Graphical Kernel System (GKS)
- GKS 3D
- PHIGS
- PHIGS+
- Computer Graphics Interface (CGI)
- Графические протоколы
- X-Window System

Периоды развития графических средств и стандартизация

1 СТАНДАРТИЗАЦИЯ В МАШИННОЙ ГРАФИКЕ

Начальный период создания и развития средств машинной графики характеризовался развитием многочисленных, иногда достаточно эффективных, графических систем, ориентированных на то или иное оборудование [Баб$_7$8, ?].Фактическиэтотпериодможноохарактеризоватьк

Первые результаты по стандартизации были получены применительно к сети ARPA в рамках работ по разработке протоколов для аппаратно и машинно-независимого представления графических данных в сети [?].

Модель работы пользователя в сети с применением графического протокола [?] приведена на рис. ??.

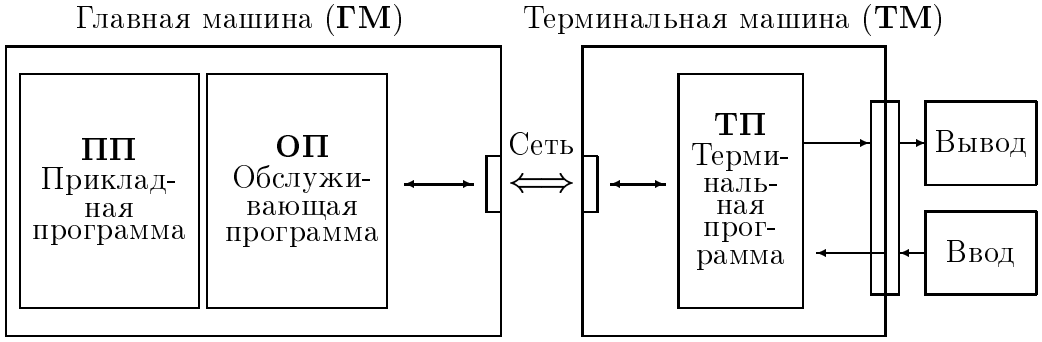


Рис. 1: Модель организации работы в сети

В начале сеанса работы пользователь располагается перед дисплеем, подключенным к терминальной ЭВМ, инициирует терминальную программу и устанавливает связь с ЭВМ сети, на которой ему будет предоставлено необходимое обслуживание. Прикладная программа главной ЭВМ при необходимости выполнить ввод/вывод использует обслуживающую программу, функцией которой является интерфейс прикладной программы с се-

тевым графическим протоколом.

Терминальная программа используется для интерпретации протокола в подходящую аппаратно-зависимую форму и для реализации функций ввода и запроса к обстановке.

Данные в сети передаются только в стандартной форме, следовательно, на передающей стороне выполняется кодирование, а на приемной — декодирование информации. Предусмотрено два уровня протокола вывода: сегментированный и структурированный форматы. В сегментированном формате изображение строится из отдельных сегментов, представляющих собой список графических примитивов (точек, линий, строк текста). ТП выполняет только перекодировку и может быть достаточно простой. В структурированном формате изображение строится из вызовов отдельных подкартин, состоящих из примитивов и вызовов других подкартин. Объем передаваемых данных уменьшается, но на ТМ, как правило, требуется программное выполнение преобразований.

Для работы с виртуальными устройствами ввода, служащими для выполнения позиционирования, ввода скалярного значения, ввода состояния кнопки (вкл/выкл), ввода строки символов, ввода времени, используется два метода: 1) запроса и получения состояния устройства ввода, например, строки символов; 2) разрешения пользователю совершить действия, приводящие к возникновению события ввода и получения на главной ЭВМ “сообщения” события.

Аппаратная независимость обеспечивается средствами опроса, который позволяет выяснить конфигурацию

и возможности используемых устройств. Адаптация к возможностям реализуется необходимыми настройками прикладной и обслуживающей программ.

После публикации [?] появился целый ряд работ, посвященных использованию идей протокола в нашей стране [?, ?, ?, ?].

1.2 Международная деятельность по стандартизации в машинной графике

Работы по протоколам послужили отправной точкой по развитию стандартизации в машинной графике. В 1974 г. в США был создан комитет по стандартизации машинной графики GSPC в ACM/SIGGRAPH. В 1975 г. в ФРГ в Институте стандартов был создан подкомитет по машинной графике — DIN-NI/UA-5.9. В 1977 г. в международной организации по стандартизации (ISO) была создана рабочая группа TC97/SC5/-WG2 “машинная графика” [?].

Важную роль в разработке методологии стандартизации машинной графики сыграла конференция в Сейлаке (Франция), организованная графическим подкомитетом WG 5.2 IFIP в 1976 г. На конференции были сформулированы и обсуждены основные условия и проблемы стандартизации. Было установлено, что основная цель стандартизации — переносимость графических систем, которая достигается стандартизацией интерфейса между графическим ядром системы (базовой графической системой), реализующим собственно графические функции, и моделирующей системой — проблемно-ориентированной прикладной программой, использующей функции графического ядра. Базовая

система должна обладать: независимостью от вычислительных систем; независимостью от языков программирования; независимостью от области применения; независимостью от графических устройств.

Структура прикладной графической системы, удовлетворяющей сформулированным требованиям, может быть представлена в виде шестиуровневой модели (рис. ??).

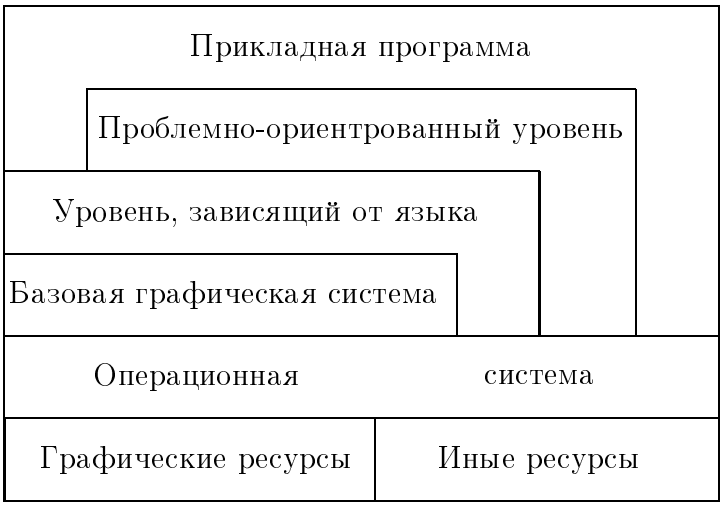


Рис. 2: Уровневая модель прикладной графической системы

Процесс преобразования информации при выполнении вывода может быть представлен состоящим из следующих этапов (рис. ??):

1. Модельные преобразования. Проблемно-ориентированный уровень из геометрических моделей отдельных объектов, задаваемых в собственных локальных системах координат, формирует описание совокупного объекта в некоторой единой (мировой) системе координат. Описание совокупного объекта подается в графическую систему.

2. Нормализующие преобразования. Графическая система переводит описание из мировой, вообще говоря произвольной, системы координат в т.н. нормализован-

ные координаты устройства, имеющие фиксированные пределы изменения координат, например, от 0.0 до 1.0.

3. Преобразования сегментов. Если графическая система предоставляет средства манипулирования отдельными подкартинами изображения (часто именуемыми сегментами), например, для независимого размещения отдельных самостоятельных частей изображения, то могут потребоваться такие преобразования.

4. Видовые преобразования. В случае 3D описания изображения и 2D устройства вывода необходимо выполнить проецирование изображения на заданную картинную плоскость. Наоборот, при 2D сцене и 3D устройстве вывода необходимо выполнить преобразование, связанное с размещением изображения. При выполнении этих преобразований, естественно, может потребоваться выполнение отсечения частей изображения. После этого этапа по сути дела готово описание изображения в некоторой аппаратно-независимой форме, пригодной для вывода на

любое устройство.

5. Преобразование рабочей станции. Для выполнения вывода на конкретное устройство необходимо преобразование данных из аппаратно-независимой формы в координаты устройства.

Процесс преобразования координатной информации при вводе от координатных устройств обратен вышеописанному преобразованию при выводе.

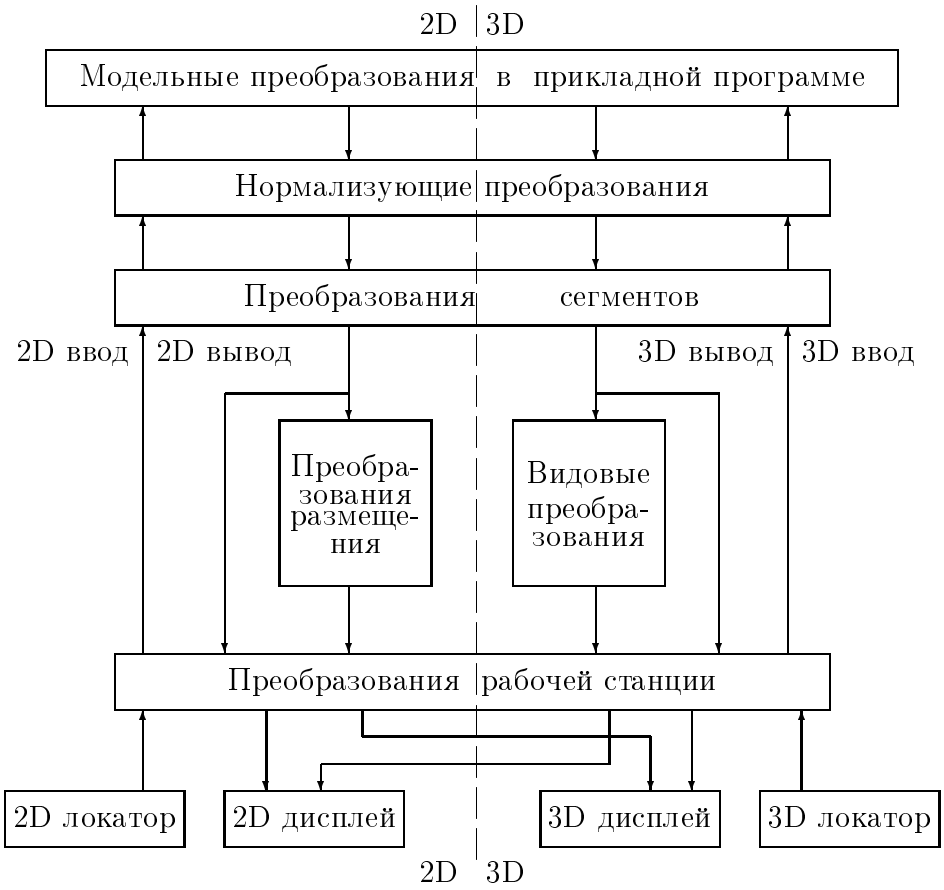


Рис. 3: Схема преобразований координатной информации в графической системе

Концептуальная модель переносимой графической системы показана на рис. ??. Штриховые линии на нем обозначают интерфейсы, при стандартизации которых может быть обеспечена переносимость.

Верхний уровень стандартизации — IGES предназначен для обеспечения мобильности компонент САПР.

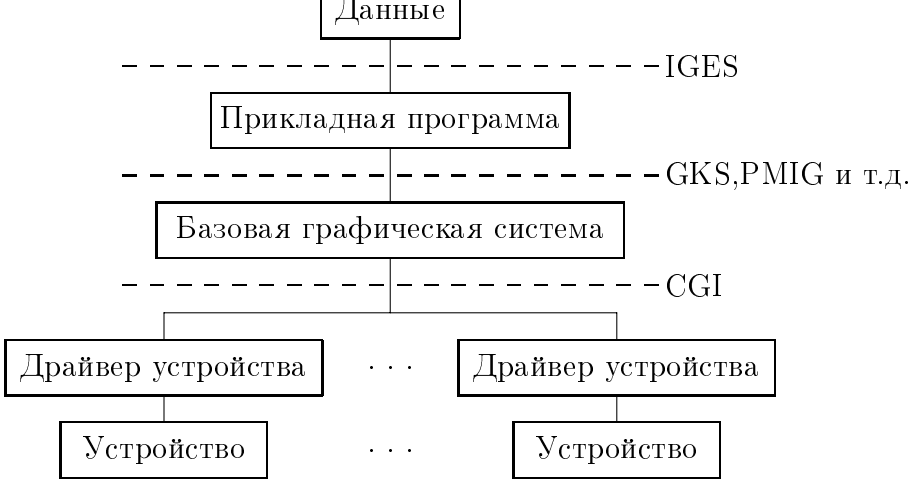


Рис. 4: Архитектура переносимой графической системы

Средний уровень стандартизации — уровень базового графического пакета (GKS) определяется выбором базовых функций системы. Этот интерфейс делает базовую графическую систему независимой от области применения.

Нижний уровень стандартизации — уровень связи с виртуальным графическим устройством (CGI) зависит от выбора примитивов ввода/вывода, являющихся абстракцией возможностей устройств. Этот интерфейс делает базовую графическую систему аппаратно-независимой.

Независимость от вычислительных систем и языков программирования обеспечивается соответствующей дисциплиной программирования и взаимодействия с системами программирования.

1.3 Деятельность ISO, IEC по стандартизации в машинной графике

Главными организациями формирующими международные стандарты в области информационной технологии являются ISO (International Organization for Standartization)

и IEC (International Electrotechnical Commission). В конце 1987 г. был сформирован первый совместный технический комитет (JTC1) ISO/IEC с целью стандартизации в области информационной технологии. Стандартизацией в машинной графике занимается 24-й подкомитет (ISO/IEC JTC1/SC24). В 1988 г. была создана постоянная советская часть этого подкомитета. Основными стандартами являются [?]:

1. GKS (Graphical Kernel System) — набор базовых функций для 2D аппаратно-независимой машинной графики.

2. GKS-3D (Graphical Kernel System for 3 Dimensions) — расширение GKS для поддержки базовых функций в 3D.

3. PHIGS (Programmer's Hierarchical Interactive Graphics System) — набор базовых функций 3D графики аналогичный GKS-3D, но в отличие от GKS-3D, ориентированной на непосредственный вывод графических примитивов, группируемых в сегменты, графическая информация накапливается в иерархической структуре данных. В целом PHIGS ориентирован на приложения, требующие быстрой модификации графических данных, описывающих геометрию объектов.

4. Языковые интерфейсы (Language bindings) — представление функций и типов данных функциональных графических стандартов в стандартизованных языках программирования.

5. CGM (Computer Graphics Metafile) — аппаратно-независимый формат обмена графической информацией. Используется для передачи и запоминания информации, описывающей изображения.

6. CGI (Computer Graphics Interface) — набор базовых элементов для управления и обмена данными между аппаратно-независимым и аппаратно-зависимым уровнями графической системы.

7. CGRM (Computer Graphics Reference Model) — модель стандартов в машинной графике, которая определяет концепции и взаимосоотношения применительно к будущим стандартам в машинной графике.

8. Регистрация — механизм регистрации стандартизуемых аспектов примитивов вывода, обобщенных примитивов, escape-функций (для доступа к аппаратным возможностям устройств) и других графических элементов.

9. Тестирование реализаций на соответствие графическим стандартам — основные цели этого проекта: специфицирование характеристик стандартизованных тестов, используемых для определения соответствия реализаций графическим стандартам, и выработка предписаний разработчикам функциональных стандартов относительно правил соответствия.

В составе 24-го подкомитета имеется 5 рабочих групп (WG):

WG1: Архитектура. Цель этой группы — развитие CGRM — модели стандартов машинной графики.

WG2: Интерфейсы прикладных программ. Стандартизация функциональных спецификаций для интерфейсов прикладных программ.

WG3: Метафайлы и интерфейсы с устройствами. Стандартизация обмена графической информацией, включая метафайл и интерфейс с устройствами.

WG4: Языковые интерфейсы. Стандартизация язы-

ковых интерфейсов для функциональных стандартов машинной графики.

WG5: Верификация, тестирование и регистрация. Разрабатывает методы и процедуры проверки соответствия и тестирования реализаций функциональных стандартов машинной графики и методов и процедур регистрации графических примитивов.

1.4 Классификация стандартов

Из рис. ?? видно, что для обеспечения мобильности программного обеспечения требуется стандартизовать:

- базовую графическую систему, т.е. стандартизовать графический интерфейс (набор базовых графических функций) — Core System, GKS, GKS-3D, PMIG, PHIGS, PHIGS+ и т.д.
- графический протокол (порядок и правила обмена информацией) — IGES, CGM и др.

Далее будут рассмотрены графические интерфейсы, являющиеся международными графическими стандартами, а затем — графические протоколы, среди которых большая часть — стандарты де-факто и только один — CGM — международный стандарт.

1.5 Core-System

Существенным этапом в области стандартизации машинной графики явилась публикация проекта стандарта CORE-SYSTEM (GSPC-77) [?], модель которой приведена на рис. ??. Главные идеи, положенные в основу системы CORE [?, ?]: разделение функций ввода и вывода; минимизация отличий между выводом на графопостроитель и интерактивный дисплей; концепция двух

координатных систем — мировой системы координат, в которой конструируется выдаваемое изображение, и приборной системы координат, в которой представляются данные для отображения; концепция дисплейного файла, содержащего приборную координатную информацию; понятие дисплейного файла сегментов, каждый из которых может независимо модифицироваться как элемент; обеспечение функций преобразования данных из мировой системы координат в приборную путем вызова видового преобразования.

В системе выделены следующие группы функций: вывода; сегментирования дисплейного файла; установления и опроса атрибутов примитивов (цвет, яркость, ширина линии и т.д.) и атрибутов сегментов (тип, видимость, указуемость и т.д.); визуализации; выполнения ввода с виртуальных устройств ввода типа указка, клавиатура, кнопка, локатор, датчик; управления и доступа к специальным аппаратным возможностям.

В 1979 г. был опубликован уточненный проект стандарта CORE-SYSTEM (GSPC-79) [?]. Кроме прочих изменений, в этой версии предусмотрена (весьма ограниченно) поддержка растровых устройств. Всего предлагалось 266 функций, так что охватывался широкий спектр применения машинной графики, начиная от пассивного вывода до интерактивных систем высокого уровня.

Ясно, что для многих приложений требуется лишь часть возможностей графпакета, все остальные, если будут присутствовать, будут приводить к неэффективности прикладной программы. Для устранения этого противоречия система разбита на три не зависящие друг

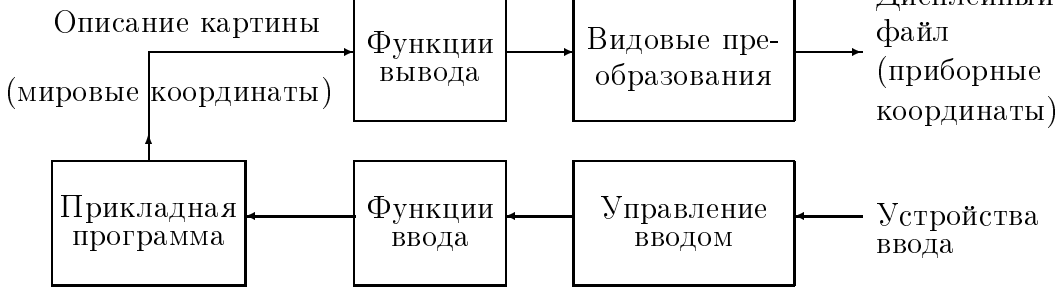


Рис. 5: Модель графической системы, положенная в основу CORE-SYSTEM

от друга группы уровней — группа уровней вывода, группа уровней ввода, группа уровней размерности. Уровни внутри группы совместимы снизу-вверх.

Группа уровней вывода включает в себя:

- базовый вывод, поддерживающий полный набор примитивов вывода, их атрибутов и операций визуализации и предназначенный для приложений, не требующих выборочного редактирования изображений;
- буферизованный вывод дополнительно к предыдущему уровню позволяет использовать сохраняемые сегменты без преобразования образа — преобразования графической информации, содержащейся в сегменте в момент выполнения вывода.

Группа уровней ввода включает:

- без ввода, т.е. применяется для пассивных приложений;
- синхронный ввод — ввод производится синхронно с работой прикладной программы, т.е. ее исполнение приостанавливается до завершения ввода;
- асинхронный ввод — ввод производится независимо от работы прикладной программы, а вводимые оператором данные накапливаются в обрабатываемой прикладной программой очереди ввода.

На последних двух уровнях поддерживаются виртуальные устройства ввода классов ЛОКАТОР, ШТРИХ, ДАТЧИК, ВЫБОР, УКАЗКА и КЛАВИАТУРА.

Группа уровней размерности включает:

• 2D — поддерживаются только 2D операции;

• 3D — дополнительно к 2D поддерживаются и 3D операции.

Предложения GSPC получили широкий отклик в виде многочисленных реализаций версий базовой системы [?, ?, ?, ?, ?].

1.6 GKS (Graphical Kernel System)

Результатом работ в ФРГ было создание системы GKS. Модель графической системы, положенная в ее основу, приведена на рис. ??. В 1979 г. GKS была принята в качестве отправной точки международного стандарта. В процессе разработки международного стандарта в исходную версию GKS был внесен целый ряд изменений, приблизивших ее к CORE-SYSTEM, но сохранивших ряд отличительных фундаментальных концепций. Основной из таких отличительных черт является введение понятия рабочей станции, представляющей собой абстракцию совокупности виртуальных устройств ввода/вывода, более полно соответствующей современной тенденции использования интеллектуальных терминалов.

По максимуму рабочая станция обладает следующими свойствами: имеет одну адресуемую поверхность вывода с фиксированным разрешением; допускается только прямоугольное поле отображения, которое не может состоять из нескольких отдельных частей; позво-

ляет задание и использование поля отображения, которое меньше максимально возможного, с гарантией того, что изображение не генерируется вне заданного поля отображения; поддерживает несколько типов линий, шрифтов текста, размеров символов и т.д. для вывода примитивов с различными атрибутами; имеет одно или больше устройств ввода для каждого класса устройств, которые независимо друг от друга могут работать в одном из трех режимов — синхронный ввод, опрос, асинхронный ввод; запоминает сегменты и обеспечивает средства для изменений сегментов и манипуляций с ними.

Ясно, что выделение рабочей станции излишне громоздко, когда приходится иметь дело с одним устройством вывода и ввода. Основная ценность введения понятия рабочей станции состоит в удобной и естественной возможности разделения аппаратно-независимой и аппаратно-зависимой частей.

Следует отметить, что в GKS столь же последовательно проводится разделение функций ввода/вывода, как и в CORE.

Набор примитивов GKS подобен набору примитивов CORE, хотя меньше и несколько более приспособлен для целей растровой графики (ломаная; полимаркер; текст; многоугольник, который может быть “залит” одним цветом, заштрихован, покрыт узором, либо может быть не закрашен за исключением границ; массив прямоугольных ячеек, закрашенных одним цветом; обобщенный примитив черчения, дающий доступ к специальным геометрическим и графическим возможностям устройств).

В отличие от CORE в GKS нет понятия текущей позиции, так что построение примитива не зависит от предыстории вычерчивания. В GKS имеется 2 способа задания атрибутов примитивов. Первый — индивидуальный способ аналогичен используемому в CORE, не зависит от рабочей станции и основан на индивидуальном задании атрибутов, которые сохраняют свое значение пока не будут изменены прикладной программой. Второй — групповой способ зависит от рабочей станции и основан на задании независимых групп значений атрибутов. Для использования требуемой группы атрибутов задается ее номер, который сохраняет свое значение пока не будет переустановлен.

Видовые операции, определяющие переход от входных (мировых) координат к физическим координатам устройства, также похожи, но в GKS, в отличие от CORE, допускается наличие не одной, а нескольких пар окно/поле вывода.

Средства сегментации GKS напоминают средства сегментации CORE, но в GKS добавлены средства манипулирования сегментами на заданной рабочей станции и средства работы с рабочей станцией специального типа — приборно-независимым хранилищем сегментов (ПНХС).

Виртуальные устройства ввода также подобны, хотя названия несколько отличаются и в GKS правила работы с устройствами лучше проработаны.

Для повышения эффективности конкретных приложений в GKS, аналогично CORE, предусмотрено разбиение на уровни. Предлагается три уровня вывода и три уровня всех остальных функций.

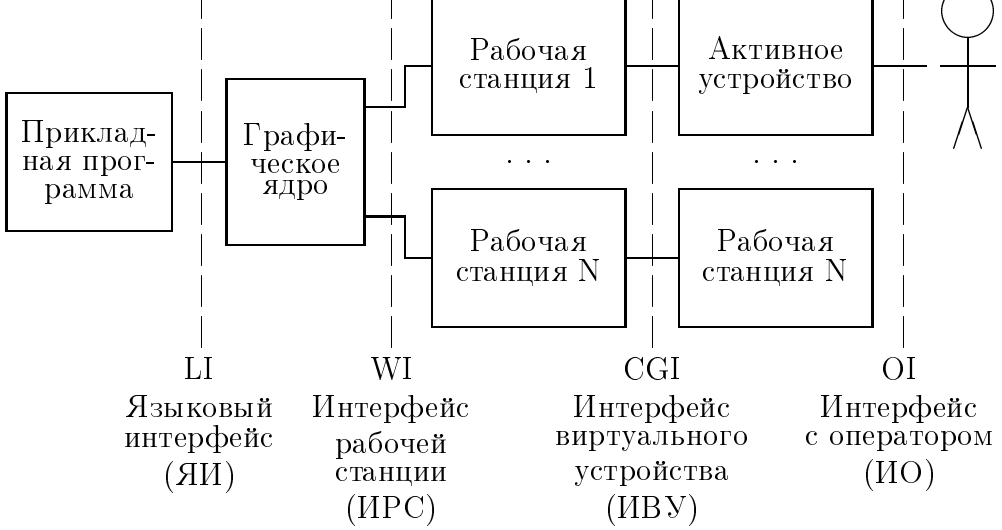


Рис. 6: Модель графической системы, положенная в основу GKS

Уровни вывода:

0: Минимальный вывод. Доступны все примитивы. Используются только групповые, немодифицируемые атрибуты примитивов. В каждый момент времени активна только одна рабочая станция вывода. Опросы параметров системы — только базовые. Доступно чтение пикселов — отдельных точек изображения из растровых устройств.

1: Базовая сегментация с полным выводом. Все возможности уровня 0. Полное управление рабочей станцией. Полные возможности вывода. Полные возможности групповых таблиц атрибутов. Одновременно могут быть активными несколько рабочих станций. Есть рабочие станции типа метафайл для сохранения/воспроизведения изображений. Может использоваться требуемое число пар окно/поле вывода. Есть базовая сегментация (без ПНХС). Доступны требуемые опросы.

2: Приборно-независимое хранилище сегментов. Все возможности уровня 1. Имеется ПНХС.

Уровни ввода:

A: Нет ввода.

B: Синхронный ввод. Существуют виртуальные устройства ввода каждого класса, работающие в режиме синхронного ввода (request).

C: Полный ввод. Все возможности ввода уровня B. Предусмотрены режимы асинхронного ввода и ввода по опросу (event и sample).

В 1985 г. GKS была принята в качестве международного стандарта [?]. В 1988 г. для стандартизованных языков программирования были приняты международные стандарты на интерфейс GKS с языками Fortran, Pascal, Ada, C [?].

1.7 GKS-3D (Graphical Kernel System for Three Dimensions)

Отличия GKS-3D от GKS заключаются в добавлении 3D функций:

- примитивов 3D вывода;
- установки атрибутов вывода (2 функции);
- поддержки 3D преобразований (9 функций);
- работы с 3D сегментами и преобразований 2D сегментов в 3D и наоборот (4 функции);
- ввода с 3D координатных устройств (10 функций);
- утилит работы с матрицами 3D преобразований (2 функции).

В целом GKS может рассматриваться как подмножество GKS-3D, т.е. 2D приложения, написанные на GKS, гарантированно исполняются в 3D окружении без каких-либо изменений.

В 1988 г. GKS-3D была принята в качестве международного стандарта [?]. В период с 1988 по 1990 гг. прорабатывались проекты стандартов ISO 8806 на интер-

фейсы GKS-3D с языками программирования. В 1991 г. завершилось их принятие в качестве международных стандартов [?].

1.8 PHIGS (Programmer's Hierarchical Interactive Graphics System)

Во многих приложениях возникает необходимость геометрического моделирования трехмерных тел (САПР машиностроительного, строительного, архитектурного и других направлений, системы автоматизации научных исследований, системы визуализации и т.д.).

Использование GKS или GKS-3D для отображения результатов моделирования предполагает, что моделирование целиком должна выполнять прикладная программа, так как эти системы ориентированы на прямой ввод/вывод и в них не предусмотрено иного манипулирования графическими данными кроме накопления в сегментах.

PHIGS [?, ?, ?, ?, ?] же комбинирует графику с техникой моделирования и представляет собой набор функций программирования графики с поддержкой быстрой модификации графических данных, описывающих геометрические соотношения объектов.

Набор примитивов вывода и их атрибутов в PHIGS тот же самый, что и в GKS-3D с добавлением примитива "annotation text relative" для пометки объектов.

Принципиальное отличие PHIGS от GKS состоит в том, что в PHIGS создание и отображение изображения разделены на две независимых фазы — занесения в централизованную структурную память (CSS — Centralized Structure Store) и отображения заданной

структуры на требуемую рабочую станцию.

Структура состоит из графических и неграфических элементов. Поддерживаются возможности создания и редактирования структур в CSS. Структура может вызывать другие структуры за счет чего создаются иерархии. Предусмотрено 9 классов структурных элементов:

примитивы вывода;
локальное преобразование моделирования;
обобщенный структурный элемент;
глобальное преобразование моделирования;
отсечение.
спецификация атр
редактиро
прикладные да
управ

Графический вывод задается в т.н. модельных координатах. Элемент “локальное преобразование моделирования” задает преобразование, которое применяется к последующим примитивам вывода.

Элемент “редактирование” определяет метку, используемую при редактировании и несущественную при отображении. Предоставлены возможности поиска метки и установки на нее указателя.

“Обобщенный структурный элемент” позволяет определить нестандартные действия.

Элемент “прикладные данные” несущественен при отображении. Обычно представляет собой указатель на прикладную базу данных.

Основное отличие между системами PHIGS и GKS-3D заключается в наличии элемента “Исполнить структуру” (Execute Structure), позволяющего при отображении исполнить структуру как часть другой. Когда такой элемент принимается интерпретатором, то текущее

состояние упрятывается и управление переключается на структуру, заданную в элементе. После завершения интерпретации вызванной структуры восстанавливается исходное состояние и интерпретация продолжается с элемента, следующего за элементом “Исполнить структуру”.

При вызове структуры она наследует глобальное модельное преобразование вызвавшей структуры. Даже если при исполнении вызванной структуры глобальное модельное преобразование будет изменено с помощью соответствующих функций, тем не менее по возврату исходное глобальное преобразование будет восстановлено.

Локальное модельное преобразование, комбинируясь с глобальным, формирует текущее модельное преобразование.

В PHIGS может быть отредактирован отдельный графический элемент структуры, что обеспечивает большую скорость модификации изображения по сравнению с GKS, в которой, как минимум, требуется перегенерация сегмента, содержащего изменяемый примитив.

Отображение в PHIGS

PHIGS — 3D система, хотя ряд функций позволяет работать в 2D. При этом все функции преобразуются в эквивалентные 3D вызовы.

В отличие от ряда подобных систем, все координатные системы в PHIGS — правые координатные системы с возрастанием координаты по оси Z при движении по направлению к наблюдателю.

При интерпретации структуры из ее элементов генерируются примитивы вывода с использованием текущего модельного преобразования. Координаты сгенерированных примитивов вывода заданы в мировой системе координат (World Coordinates — WC).

Отображение в PHIGS выполняется в две стадии. На первой, называемой View Orientation (ориентация отображения), координатная система изменяется к необходимой для отображения выводимого объекта. На второй стадии, называемой View Mapping (преобразование отображения), объект преобразуется в аппаратно-независимые нормализованные координаты устройства, готовые для вывода на каждой открытой рабочей станции. Проецирование выполняется на этой стадии.

View Orientaion изменяет мировую систему координат к системе координат точки наблюдения (View Reference Coordinate system — VRC). Суть состоит в переносе начала координат в точку фактического наблюдения объекта и с направлением осей согласно требуемой проекции.

View Mapping более сложна и зависит от выбранного типа проекции — “параллельной” или “перспективной”. При параллельной проекции объем видимости определяется как параллелепипед преобразуемый в порт проекции, который представляет собой прямоугольный параллелепипед в нормализованных координатах проекции с ребрами параллельными осям. При перспективной проекции объем видимости — конус.

Для каждого типа проекции плоскость проекции — окно, определяемое X-Y размерами параллелепипеда или конуса. Плоскость проекции перпендикулярна Z-

оси. Для параллельных проекций угол, который образуют проекторы с плоскостью проекции, определяется точкой центра проекции (Projection Reference Point — PRP). Проекторы параллельны линии из PRP в центр окна на плоскости проекции. При перспективной проекции все проекторы проходят через PRP.

Не требуется, чтобы точка проецирования (PRP) лежала на оси Z. В общем случае все типы проекций могут быть получены путем ориентации объекта относительно VRC осей и посредством размещения PRP относительно VRC-осей и плоскости проекции.

Для многих применений требуется одновременное наличие нескольких видов объекта. В PHIGS возможно определение множества видов одного или различных объектов. Для каждого вида SET VIEW REPRESENTATION (установить представление вида) задает как он будет показан на требуемой рабочей станции. Отсечение производится для создания части пространства нормазизованных координат проекции (Normalized Projection Coordinates — NPC), потенциально видимой на рабочей станции для этого вида.

Преобразования окно/порт рабочей станции близки к GKS в определении расположения пространства нормализованных координат проекции на рабочей станции и в одинаковости коэффициентов масштабирования вдоль направлений X и Y.

1.9 PHIGS+

PHIGS+ [?, ?, ?] — расширение PHIGS, имеющее дополнительные функциональные возможности для приложений, требующих учета освещенности, раскраски

(интерполяции цветов по поверхности), а также дополнительные возможности по управлению отображением и новые примитивы для поддержки эффективного описания сложных поверхностей. Эти расширения были сформулированы как поправка к существующим частям 1–3 стандарта PHIGS и введением новой части ь 4 стандарта.

Поддержка освещенности и раскраски основана на предоставлении средств задания позиции источника света и наличии примитивов “с данными”, задающими вектора нормалей и цвета вершин.

Предоставлена возможность задания в структуре “ограничивающего бокса”. Когда он обрабатывается, то устанавливаются флаги отсечения (если ограничивающий бокс полностью вне области вывода) и отбраковки (если ограничивающий бокс меньше заданного размера). Этот дополнительный элемент структуры предназначен для управления пропуском исполнения структуры, образ котрой находится вне области вывода, и позволяет повысить скорость отображения.

1.10 CGI (Computer Graphics Interface)

Это стандарт ISO [?] на интерфейс между аппаратнонезависимой частью графического программного обеспечения (базисной графической системой) и аппаратнозависимой (драйверами). Этот интерфейс ранее (в рамках ANSI) назывался интерфейсом виртуального устройства.

Для эффективного использования аппаратных возможностей современных графических устройств набор

функций CGI перекрывает аппаратно-реализуемые возможности и включает в себя следующие функции:

- управление устройством,
- вывод графических примитивов,
- изменение графических атрибутов,
- сегментация изображений,
- графический ввод,
- растровые операции.

Отличительными особенностями CGI (по сравнению со стандартами на интерфейс базисной графической системы) являются следующие: расширенный набор графических примитивов, одноступенчатое преобразование координат, увеличенное количество логических устройств ввода, наличие растровых операций. В целом набор функций CGI достаточно удобен для создания надстроенного над ним графического программного обеспечения. Последнее позволяет эффективно создавать на основе CGI различные базисные графические системы.

Следует отметить, что стандарт на интерфейс виртуального устройства ориентирован в первую очередь на унификацию способа взаимодействия с различными графическими устройствами и предназначен для системных, но не прикладных программистов. Он был утвержден после появления множества проектов по стандартизации программных интерфейсов базисной графической системы.

Анализ применяемых в настоящее время графических протоколов и проектов по их стандартизации позволяет выделить протоколы следующих типов:

- аппаратно-зависимые графические протоколы или команды графических устройств,
- аппаратно-независимые графические протоколы или метафайлы,
- прикладные графические протоколы,
- растровые графические файлы.

1.11.1 Аппаратно-зависимые графические протоколы

Аппаратно-зависимые графические протоколы разрабатываются фирмами, производящими графическое оборудование. Они представляют собой последовательность команд для построения изображений на устройствах выпускаемых данной фирмой. Для интерпретации таких протоколов не требуется дополнительных ресурсов если используется соответствующее устройство. Поэтому, такие протоколы могут успешно применяться в распределенных системах при отсутствии локальной ЭВМ.

Вопрос о поддержке тех или иных аппаратно-зависимых графических протоколов определяется составом используемого оборудования. Целесообразно, чтобы центральная ЭВМ обеспечивала возможность генерации команд для наиболее распространенных графических устройств. В настоящее время значительная часть производящейся в мире графической аппаратуры работает с протоколами TEKTRONIX, REGIS и HPGL. Поддержка этих протоколов обеспечивается также в наиболее распро-

страненных зарубежных программных продуктах.

Протокол TEKTRONIX

Разработан одноименной фирмой, выпускающей графические дисплеи. Ввиду широкой распространенности устройств этой фирмы другие разработчики графической аппаратуры часто обеспечивают режим совместимости с TEKTRONIX'ом в своих устройствах. Поддержка этого протокола производится также в некоторых эмуляторах терминала (например, VTERM, ST240, TEEMTALK) работающих на персональных компьютерах типа IBM PC.

Существуют две разновидности протокола TEKTRONIX, соответствующие дисплеям серии 4010/12/14 и дисплеям серий 41XX, 42XX, 43XX. Дисплеи 4010/12/14 это дисплеи на запоминающей трубке с минимальным набором графических команд. Дисплеи серии 41XX и выше это цветные растровые дисплеи, работающие с более развитым графическим протоколом, который предусматривает следующие основные группы графических команд:

- команды построения векторных примитивов,
- команды работы с растровыми изображениями,
- команды управления сегментацией изображения,
- команды задания цветовых и геометрических атрибутов,
- команды графического ввода,
- команды управления плоскостями вывода,
- команды выполнения видовых преобразований,

• команды определения символов (графических образов).

Кроме перечисленных графических функций протоколы серии TEKTRONIX 41XX и старше включают множество функций управления алфавитно-цифровым режимом, каналом передачи данных и дополнительными внешними устройствами (hardcopy, планшет, диск).

Для представления команд используется символьное кодирование с использованием служебных символов в имени команды (чаще всего — символа Esc), что затрудняет чтение операторов человеком.

Протокол REGIS

Разработан для дисплеев серии VT (240 и выше). С этим протоколом работают также персональные компьютеры фирмы LabTam и ряд графопостроителей различных фирм. Поддержка протокола REGIS обеспечивается в некоторых эмуляторах терминала на IBM PC (VTERM, ST240). По функциональным возможностям протокол REGIS заметно уступает протоколу TEKTRONIXa. В частности, в нем гораздо беднее набор растровых операций, задания атрибутов построений, средств графического ввода и управления плоскостями вывода, полностью отсутствуют возможности сегментации изображения, выполнения видовых преобразований, определения символов.

Протокол HP-GL

Графический протокол HP-GL (язык описания данных Graphic Language) разработан фирмой Hewlett Packard в 1976 г. и поддерживается множеством других фирм,

выпускающих графопостроители. В настоящее время используется версия HP-GL/2. Операторы языка содержат символьное имя команды и несколько параметров, также подготовленных в печатном текстовом виде. Всего в языке 88 операторов, разбитых на 9 функциональных групп. Ядро языка содержит 5 групп из 55 операторов, которые должны поддерживаться на всех устройствах. Оставшиеся 3 группы из 33 операторов являются специфичными для некоторых из устройств. В целом, благодаря использованию явного текстового представления, язык легко читается и интерпретируется.

1.11.2 Языки описания страниц

Любая страница может быть описана как просто пиксельный массив, но это практически неприемлемо. Язык описания страниц должен описывать любой текст и графику на высоком уровне в терминах абстрактных графических элементов.

Выполнение вывода с использованием языка описания страниц идет в две стадии:

1. Приложение генерирует аппаратно-независимое описание на языке описания страниц.

2. Программа, управляющая некоторым растровым устройством вывода, интерпретирует описание и отображение его на устройство.

Эти две стадии могут быть выполнены в разное время и в разных местах.

Примитивы вывода выдаются на растровое устройство вывода процессом, называемым преобразованием сканирования (растеризация).

Особое место среди графических языков высокого уровня занимает интерпретируемый язык описания страниц PostScript [?, ?], разработанный фирмой Adobe и используемый не только для описания и построения изображений, но и качестве высокоуровневого аппаратно-независимого протокола обмена между компонующей и отображающей системами.

В первую очередь PostScript — это общий язык программирования с встроенными мощными графическими примитивами. С другой стороны, это язык описания страниц, который включает возможности языка программирования, и используется для связи с электронными печатающими устройствами.

На PostScript'е можно описывать любые графические формы, двухуровневые изображения и печатаемые формы. Для построения изображений (в том числе и всевозможных шрифтов) в языке PostScript предоставляется возможность управления каждой точкой печатающего устройства.

Естественно, что вследствие наглядности PostScript, как и другие языки программирования неоптимален в смысле минимальности кодирования информации. Поэтому его использование в качестве графического протокола представляется нецелесообразным. Однако он становится незаменим при передаче тексто-графических документов, предназначенных для воспроизведения на печатающих устройствах с высоким разрешением (например, лазерных принтерах).

Аппаратная независимость достигается тем, что по-

строение изображения ведется в пользовательской системе координат с помощью обычных графических примитивов и описание изображения не содержит никакой информации об устройстве отображения.

В языке предусмотрен ряд типов данных — числа, строки, одномерные массивы, словари (таблицы, задающие соответствие между ключом и значением). Элементы массивов могут быть различных типов. Примитивы управления включают в себя условия, циклы и процедуры, которые могут вызываться рекурсивно.

Операторы (арифметические, логические, графические и управления) записываются в постфиксной записи и манипулируют со стеком типа LIFO.

В язык встроены следующие изобразительные возможности:

- изображения строятся из отрезков линий, дуг и кубических кривых;
- примитивы могут быть выведены линиями требуемого вида, закрашены любым цветом или использоваться для задания области отсечения для других графических элементов;
- текст полностью интегрирован с графикой, символы как стандартных шрифтов, так и определенных пользователем, рассматриваются как графические образы, которые обрабатываются графическими операторами языка;
- 2D общая координатная система поддерживает обычные линейные преобразования и их комбинации (сдвиг, поворот, масштабирование, отражение и т.д.);

- как построенные графическими операторами, так и естественные изображения, введенные, например, со сканера, могут иметь требуемые разрешение и динамический диапазон.

PostScript стал стандартом "де-факто" и получил чрезвычайно широкое распространение как язык описания страниц для лазерных принтеров и других устройств с высоким разрешением, его интерпретаторы входят в состав контроллеров растровых принтеров многих типов.

Языки описания страниц, близкие по возможностям к PostScript, разработаны также фирмами Xerox (язык Interpress) и Imagen (язык DDL).

На основе расширения языка PostScript фирмой Sun Microsystems разработана система NEWS [?] (the Network extensible Window System) для управления окнами в сети.

Язык PCL

Несколько версий языка описания страниц Printer Communication Language (PCL) было разработано фирмой Hewlett-Packard для вывода на лазерный принтер рисунков и текстов с использованием различных шрифтов. В версии PCL5 имеется 64 оператора, разбитых на 10 функциональных групп. Все операторы начинаются с символа Esc (шестнадцатиричный код 01BH) и содержат один или несколько последующих символов. Символьное кодирование, используемое в PCL, менее приспособлено для чтения человеком, чем явное текстовое кодирование, используемое в языке PostScript, но значительно компактнее последнего.

Аппаратно-независимый графический протокол или метафайл представляют собой процедурное описание изображения в функциях виртуального графического устройства. Он обеспечивает возможность запоминать графическую информацию единым образом, передавать ее между различными графическими системами (в том числе работающими на различных ЭВМ) и интерпретировать информацию для вывода на различные графические устройства. Для интерпретации метафайла требуется локальная ЭВМ, выполняющая эмуляцию не реализованных в аппаратуре функций и кодирование в команды конкретных устройств.

В настоящее время в мировой практике наиболее активно поддерживаются стандартизованные аппаратно-независимые протоколы NAPLPS, GKSM, CGM и WMF — стандарт де-факто фирмы Microsoft на метафайл.

NAPLPS — North American Presentation Level Protocol Syntax

Это американский стандарт на представление графических данных в сетях VIDEOTEX (имеется также европейский аналог этого стандарта — SET). Основными требованиями при разработке этого протокола были следующие:

- возможность передачи графической информации в потоке буквенно-цифровых данных,
- минимальность объема передаваемых графических данных,

- минимальность усилий для интерпретации и возможность вывода изображений на простейшие графические устройства.

Обеспечение этих требований привело к тому, что был разработан эффективный способ упаковки графической информации в 7-ми или 8-битные коды ASCII. Эти же требования привели к ограничению функциональных возможностей протокола, что не позволяет получить высокое качество изображений при использовании современных графических устройств.

GKSM — Graphical Kernel System Metafile

Это de-facto стандарт на графический метафайл в рамках базисной графической системы GKS (приложение E к стандарту GKS). По функциональным возможностям этот протокол полностью соответствует системе GKS со всеми ее достоинствами и недостатками. Поэтому, он легко интерпретируется в системах, соответствующих стандарту GKS.

Недостатком GKSM-протокола (равно как и системы GKS) является, например, минимальный набор стандартизованных графических примитивов (их всего 5), что не дает возможности эффективного использования современных интеллектуальных графических устройств и увеличивает объем передаваемой информации. Возможность же использования обобщенного графического примитива или ESCAPE-механизма (предназначенных для нестандартных функций) не обеспечивает однозначность интерпретации графических данных при передаче их между различными системами.

Кодирование в GKSM текстовое, что позволяет читать (просматривать, редактировать) метафайл но требует большего объема чем символьное кодирование применяемое в NAPLPS или CGM.

CGM — Computer Graphics Metafile

Это стандарт ISO [?] на графический метафайл. Функционально этот протокол соответствует стандарту на интерфейс виртуального устройства CGI (Computer Graphics Interface), предназначенного для обеспечения связи различных графических систем с различным графическим оборудованием и являющегося обобщением текущего уровня развития графического программного и аппаратного обеспечения. По этой причине протокол CGM может совмещаться с различными программными системами и позволяет наиболее эффективно использовать возможности имеющихся графических устройств.

В CGM предусмотрены три способа кодирования — символьное, двоичное и текстовое. Символьное кодирование (по идеологии близкое к принятому в NAPLPS) достаточно компактно и предназначено для хранения и транспортировки графической информации. Двоичное кодирование требует минимальных усилий по генерации и интерпретации и предназначено для внутрисистемного использования. Текстовое кодирование наиболее наглядно и обеспечивает возможность визуального просмотра и редактирования графических файлов. При необходимости в рамках графической системы могут быть предусмотрены соответствующие перекодировщики.

DXF

Это формат графических файлов с которыми работает система AutoCad. Его следовало бы отнести к категории аппаратно-зависимых протоколов, но (в настоящее время) нет ни одного устройства понимающего этот формат. Использование этого формата целесообразно в том случае, когда есть необходимость использования графических возможностей системы Autocad.

WMF — Windows Metafile Format

В системе Windows фирмы Microsoft для сохранения и последующего использования цветных изображений используется свой формат метафайла. В WMF используется единственный способ кодирования — двоичный, который, как это отмечалось выше, наиболее компактен и обеспечивает наибольшие скорости упаковки и воспроизведения, но неудобен для просмотра и анализа человеком. Метафайл содержит заголовок и собственно описание изображения в виде записей GDI (Graphical Device Interface) функций. Всего предусмотрено три варианта метафайла — стандартный, размещаемый (placeable) и буферный (clipboard). Отличия вариантов состоят только в разных структурах заголовков. В составе Windows предусмотрены функции для создания и проигрывания метафайлов и манипулирования ими.

* * *

Перечисленные аппаратно-независимые графические протоколы ограничиваются 2-мерными графическими

примитивами, что не позволяет использовать ресурсы локальной ЭВМ для сокращения объема передаваемой информации при работе с 3D изображениями. Поэтому, представляется целесообразной разработка двухуровневого графического протокола согласованного с CGM и обеспечивающего возможность передачи 3D объектов. Сокращение нагрузки на линию связи будет происходить в данном случае не только за счет повышения семантической насыщенности передаваемой информации, но и за счет возможности получения на локальной ЭВМ множества изображений с различными параметрами визуализации (например, множество изображений пространственного объекта с различных точек зрения).

1.11.4 Проблемно-ориентированные протоколы

Прикладные графические протоколы это объектно — ориентированные протоколы передачи данных между прикладными системами. Они наиболее компактны (вследствие высокой семантической насыщенности), допускают свободу в выборе различных способов графического представления, но требуют большей мощности локальной ЭВМ для интерпретации. Прикладные протоколы стандартизованы пока только для САПР машиностроения и электроники. В качестве примеров можно привести следующие стандарты:

- IGES (Initial Graphics Exchange Specification),
- STEP (STandard for the Exchange Product Model Data),
- MAP (Manufacturing Automation Protocol),
- VDAFS (Sculptured Surface Interface),
- EDIF (Electronic Design Interchange Format).

Основные трудности, связанные с разработкой протоколов этого уровня, состоят в том, что во многих областях применения до сих пор не унифицированы основные объекты (в том числе графические) и операции над ними. Для работы в этом направлении потребуются объединенные усилия проблемных специалистов, математиков и системных программистов в области баз данных, машинной графики, телекоммуникаций и т.д.

1.11.5 Растровые графические файлы

С появлением и широким распространением персональных ЭВМ, использующих растровые дисплеи и устройства документирования (лазерные и струйные принтеры и т.д.), для целей компактного хранения и транспортировки графической информации стали активно применяться различного рода растровые графические файлы. Используется более десятка различных типов растровых графических файлов. К наиболее известным относятся:

- TIFF (Tag Image File Format),
- GIF (Graphics Interchange Format),
- PIC,
- PCX,
- MAC (MacPaint),
- BMP (Bitmap).

TIFF (Tag Image File Format)

Разработан корпорациями Aldus и Microsoft совместно с некоторыми фирмами, производящими сканеры и принтеры. Этот формат поддерживается целым рядом

систем подготовки документации и является наиболее реальным претендентом на стандарт для хранения и транспортировки растровых изображений.

Основной концепцией формата TIFF является цветовая модель изображения. Под этим понимается набор характеристик изображения, определяющих способ представления цвета. Стандартизованы следующие цветовые модели:

- двух-уровневое изображение (bi-level image);
- монохромное изображение (gray-scale image);
- индексированное цветное изображение (paletted color image);
- полное цветное изображение (full RGB image).

TIFF является открытым форматом и позволяет создать любую модель изображения. Естественно, что выбор требуемой модели определяется решаемой задачей. Например, двух-уровневая модель наиболее удобна в системах подготовки документации. Индексированное цветное изображение совместимо с форматом хранения графической информации в наиболее распространенных в настоящее время растровых графических дисплеях.

Помимо информации о модели изображения формат TIFF содержит метрические характеристики, а именно: размеры изображения, плотность (количество пикселов на единицу длины), с которой создавалось изображение. Эти характеристики особенно полезны в системах подготовки документации. TIFF не накладывает практически никаких ограничений на параметры изображения. Так, например размеры изображения могут достигать 4 миллиардов. Количество битов на пиксел

ограничено этим же числом.

Формат TIFF позволяет хранить в одном файле любое количество изображений. Кроме того, есть возможность хранить несколько копий одного изображения с различными характеристиками. Так, например, можно иметь несколько вариантов изображения, отличающихся различной плотностью, что полезно опять же в издательских системах для работы с несколькими принтерами.

В формате TIFF имеется возможность упаковывать изображение различными методами. В том числе изображение может храниться и в неупакованном виде, что представляется удобным, так как, например, при создании изображения важен произвольный доступ к любому элементу изображения за достаточно малое время. Одним из методов кодирования является LZW (Lempel, Ziv & Welch), который дает высокий коэффициент сжатия.

GIF (Graphics Interchange Format)

Разработан в CompuServe Incorporation для хранения и транспортировки растровых изображений. GIF позволяет содержать в одном файле несколько изображений, не связанных между собой.

По сравнению с форматом TIFF он более прост. В GIF используется наиболее распространенная модель изображения — индексированное цветное изображение. Хотя это не мешает хранить в нем изображения двухуровневой и монохромной моделей. Это реализуется путем соответствующей настройки таблицы цветности. Модель полноцветного изображения с хранением для ка-

ждого пиксела компонент R, G и B принципиально не вписывается в формат GIF, так как существует ограничение на количество битов в пикселе — 8.

Достоинством формата GIF является наличие стандартизованного протокола передачи GIF-изображения по линии связи. Формат GIF, в отличие от TIFF, организован по принципу последовательного доступа. Т.е. он не содержит оглавлений или каких-либо ссылок внутри себя. Это делает его удобным при передаче изображений в распределенной графической системе.

GIF использует единственный метод кодирования изображений — LZW. Это свойство следует отнести к недостаткам формата, так как использование одного метода кодирования ограничивает область его применения. Кроме того, следует отметить, что не существует графических редакторов, понимающих формат GIF.

ZSoft (PCX)

Формат распространен на IBM PC и используется в графических редакторах (Paintbrush, EgaPaint) и системах подготовки документации (Ventura Deck Top Publisher, First Publisher).

В PCX используется очень неэффективный метод кодирования, он дает низкий коэффициент сжатия. Однако время, используемое на кодирование/декодирование практически равно времени кодирования без всякой упаковки. Это дает преимущества при использовании этого формата в интерактивных системах с быстрой сменой изображений.

MacPaint (MAC)

Является основным форматом в системах подготовки документации и графических редакторах на персональных компьютерах фирмы Apple (Macintosh, Lisa).

BitMap (BMP)

Является одним из основных форматов представления растровых изображений в системе Windows. Файл имеет достаточно простую структуру и сохраняет единственное изображение с одним, четырьмя, восемью и двадцатью четырьмя битами на пиксел. Одно-, четырех- и восьмибитное представления соответствуют индексированному цветному изображению. Для таких изображений в заголовке BMP-файла хранится таблица цветности. Изображение может быть сжато с использованием кодированием длин серий для четырех- и восьмибитных изображений.

Наряду с перечисленными имеется еще ряд форматов для хранения растровых файлов, используемых в отдельных системах, но не получивших широкого распространения. Их применение может оказаться целесообразным в случае использования соответствующих систем.

1.12 X Window System

В отличие от большинства других предложений по стандартизации, таких как, например, GKS, система X спроектирована и реализована небольшой группой разработчиков из Массачусетского технологического института в рамках проекта Athena, спонсорами которого были фирмы DEC и IBM.

Система X — программное окружение для программистов и пользователей прикладного программного обеспечения инженерных рабочих станций и представляет собой целостную систему, в рамках которой синтезированы предложения по управлению окнами, функциональному и языковому интерфейсам, аппаратно-независимому протоколу и программному обеспечению рабочей станции.

Основой X является "базовая оконная система". Полное X-окружение надстроено над ней в виде уровней (рис. ??).

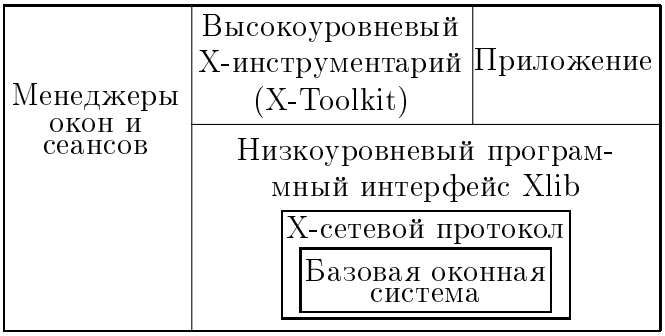


Рис. 7: Структура X

Базовая оконная система взаимодействует с окружающим миром только через сетевой интерфейс. Какого-либо специального привилегированного доступа к ней не предусмотрено.

Архитектура X-системы базируется на модели клиент-сервер (рис. ??). Клиент — прикладная программа. Сервер — программа, вызванная на компьютере, к которому физически подключен дисплей. В ней сосредоточена вся аппаратная зависимость.

Основные критерии, которыми руководствовались в разработке X-системы:

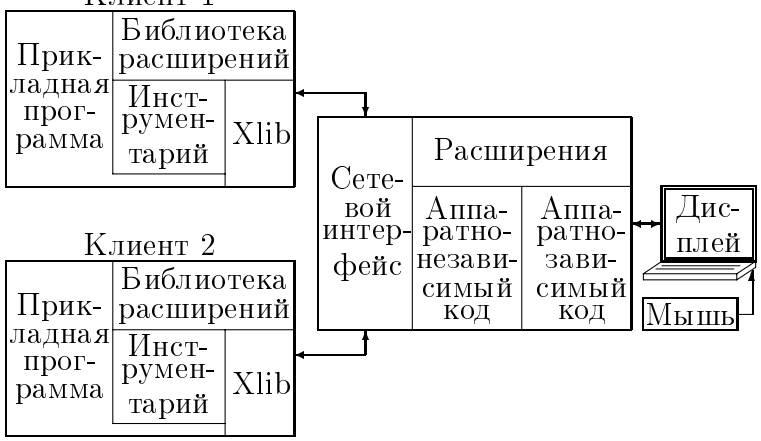


Рис. 8: Архитектура X-системы

- **транспортабельность — система должна быть применима к различным дисплеям;**
- **аппаратная независимость приложений;**
- **сетевая прозрачность — приложения могут быть вызваны как локально, так и на удаленных ЭВМ без каких-либо их изменений и без необходимости информирования об отличиях. Должны быть поддержаны различные сетевые протоколы;**
- **конкурирующие приложения — дисплей может делиться между несколькими приложениями. Возможен вывод из нескольких приложений как в единственное, так и в несколько окон;**
- **интерфейсы с приложениями и управление окнами — X не предписывает какого либо одного способа работы с окнами, например, только перекрывающимися;**
- **перекрытия окон — ввод/вывод должны быть возможны для полностью или частично затененных окон.**

* * *

Одно из основных отличительных свойств X-системы

— работа одного сервера с несколькими клиентами и наоборот одного клиента с несколькими серверами представляется весьма удобным и естественным. Подобный подход использовался, например, в системах ДИГРАФ.К и ДИГРАФ.Р [?].

Другое же основополагающее положение, что за перечерчивание окон, получивших экспозицию на экране, несет ответственность прикладная программа, приводит к необходимости иметь высокоскоростные каналы связи (например, Ethernet). Другими словами, за сравнительно малые результаты приходится платить несуразно высокую цену. Представляется целесообразным иметь управление окнами на терминальной ЭВМ, возможно и с какими-либо прагматическими ограничениями на возможности такой локальной системы управления окнами.

1.13 Выводы

В машинной графике широко распространилось понимание необходимости стандартизации, которая позволяет обеспечить переносимость пакетов прикладных программ, унифицировать графические методы работы, углубить их понимание и практического использования, ставить задачи перед разработчиками аппаратуры.

В настоящее время работы по стандартизации, в основном, сосредоточены на узком фронте специфицирования некоторого минимального набора “базисных” функций с одновременным стремлением к многофункциональности пакетов графических подпрограмм. Следует ожидать, что дальнейшее продвижение стандарти-

зации будет идти по пути повышения ее функционального уровня в определенных, сформировавшихся областях приложений.

В целом, текущее состояние работ по стандартизации машинной графики — необходимый, но пока первый шаг в этой части создающейся на наших глазах индустрии программного обеспечения [?].

Наряду с положительными аспектами стандартизации следует отметить и определенные минусы, имеющие общий характер.

1. Стандартизация всегда означает затверждение некоторого определенного уровня достижений и понимания, тем самым в определенной степени тормозится развитие новых, нестандартных технических средств и программного обеспечения. Особенно, если учесть то, что первой строчкой наших стандартов является: “несоблюдение стандарта преследуется по закону”.

2. Стремление покрыть широкий спектр применений, начиная от пассивного вывода до высокоинтерактивных приложений, несмотря на наличие уровней, все-таки приводит к громоздкости и набора средств, и структуры данных и, естественно, программного кода.

3. Стремление к легкой адаптируемости влечет за собой чрезвычайно большое количество средств запроса к обстановке (в GKS — 75 функций из общего числа 185, т.е. более 40%). Такое количество несомненно избыточно для многих конкретных приложений. Не случайно поэтому, например, еще в 1987 г. темой одной из дискуссий на Всесоюзной школе-семинаре по “Информатике и интерактивной компьютерной графике” (Цахкадзор, 16–20 марта 1987 г.) было: “Стандартизация — закон

или методология, тормоз или ускорение".

Важно отметить, что в предложениях по стандартизации наряду со стремлением к многофункциональности пакетов очевидно и стремление к минимизации набора стандартизованных примитивных функций, что, вообще говоря, неверно для конкретной области приложений. В последнем случае повышение функционального уровня стандартизации обеспечит как легкость изучения, так и легкость адаптации, которая важна ведь не вообще, а в каждом случае в некотором конкретном классе приложений. Практика написания прикладных систем показывает, что для повышения эффективности прикладных программ требуется набор различных функциональных возможностей из различных, предлагаемых стандартами уровней.

В этой связи, интересным представляется решение, предложенное в [?], положенное в основу графпакета АТОМ. Система машинной графики представляется в виде совокупности пяти сравнительно слабо связанных подмножеств: средств формирования изображений; средств промежуточного хранения информации; средств ввода; средств преобразований изображений; средств управления графическими устройствами.

Требуемая конфигурация графической системы собирается из отдельных модулей, объединяемых в конвейер [?]. Конвейер собирается либо статически — на этапе проектирования программы, либо динамически, в процессе ее исполнения. Передача данных в конвейере осуществляется через единый межмодульный интерфейс.

Курс “Компьютерная графика”

1. **Цель курса**
2. **Организация занятий**
3. **Части курса**
   - **вводный курс;**
   - **основные алгоритмы;**
   - **архитектуры графических систем.**
4. **Результаты и контроль**
5. **Литература**

Литература

1. Вельтмандер П.В. Введение в машинную графику: Учеб. пособие/ Новосиб. ун-т. Новосибирск, 1995. 77 с.
2. Вельтмандер П.В. Машинная графика. Вводный курс:Учеб. пособие в электронном виде.
3. Вельтмандер П.В. Машинная графика. Основные алгоритмы: Учеб. пособие в электронном виде.
4. Вельтмандер П.В. Машинная графика. Введение в архитектуры графических систем: Учеб. пособие в электронном виде.
5. Роджерс Д. Алгоритмические основы машинной графики. Пер. с англ. М.: Мир,1989. 512 с.
6. Павлидис Т. Алгоритмы машинной графики и обработки изображений. Пер. с англ. М.: Радио и связь, 1986.
7. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики: В 2-х книгах. Пер. с англ. М.: Мир, 1985.
8. Гилой В. Интерактивная машинная графика. Пер. с англ. М.: Мир, 1981.
9. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики. Пер. с англ. М.: Мир, 1976.
10. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. М.: “ДИАЛОГ–МИФИ”, 1995. 228 с.
11. Donald Hearn, M. Pauline Baker. Computer Graphics. Prentice Hall, 1994. 652 p.

Вводный курс

1. **История, предмет, приложения МГ**
2. **Зрительный аппарат человека**
3. **Физические принципы формирования оттенков**
4. **Цветовые модели**
5. **Печатающие устройства**
6. **Графопостроители**
7. **Электронно-лучевые трубки**
8. **Дисплеи с произвольным сканированием луча**
9. **Растровые дисплеи**
10. **Другие типы дисплеев**
11. **Дисплеи для IBM PC**
12. **Устройства ввода**
13. **Нетрадиционные устройства**

Основные алгоритмы

1. **Координаты и преобразования**
2. **Генерация векторов**
3. **Улучшение качества изображений**
4. **Генерация окружности**
5. **Генерация кривых**
6. **Заполнение многоугольника**
7. **Заливка области с затравкой**
8. **Отсечение отрезков**
9. **Отсечение многоугольника**
10. **Структуры данных**
11. **Геометрическое моделирование**
12. **Удаление скрытых линий и поверхностей**
13. **Реалистичное представление сцен**
14. **Фрактальная геометрия**

Архитектуры графических систем

1. **Интерактивные системы машинной графики**
2. **Архитектура графических рабочих станций**
3. **Стандартизация в машинной графике**
4. **Системы управления пользовательским интерфейсом**
5. **Визуализация в научных исследованиях**
6. **Оценка производительности графических систем**

История технических средств

- **1950, МТИ, машина Ураган-1 (Whirlwind 1), вывод на ЭЛТ;**
- **конец 60-х — дисплей на запоминающей трубке;**
- **1966 — плазменная панель;**
- **середина 70-х — дешевая растровая графика;**
- **1968, ВЦ АН СССР, машина БЭСМ-6, первый отечественный растровый дисплей;**
- **1974, первый отечественный серийный векторный дисплей Дельта (ИАиЭ — разработчик, ИПФ и завод Луч — запуск в серию).**

Интерактивные вычисления

1959, Стречи на конференции ЮНЕСКО по обработке информации предложил режим разделения времени.

Этапы развития

60–80-е годы — основные алгоритмы
отсечение, генерация примитивов, закраска узорами, реалистичное представление сцен (удаление невидимых линий и граней, трассировка лучей, излучающие поверхности).

90-е годы — превращение из научного направления в один из важнейших инструментов современной цивилизации.

Области машинной графики

1. **Изобразительная машинная графика.**
2. **Обработка и анализ изображений.**
3. **Перцептивная графика**
   (анализ сцен).
4. **Когнитивная машинная графика (способствующая познанию, МГ для научных абстракций).**

Изобразительная машинная графика

Объекты: синтезированные изображения.

Задачи:

- построение модели объекта и генерация изображения;
- преобразование модели и изображения;
- идентификация объекта и получение требуемой информации.

Обработка и анализ изображений

Объекты: дискретное, числовое представление фотографий.

Задачи:

- повышение качества изображения;
- оценка изображения — определение формы, местоположения, размера и других характеристик требуемых объектов;
- распознавание образов — выделение и классификация свойств (обработка аэрокосмических снимков, ввод чертежей, системы обнаружения и наведения).

Перцептивная графика (анализ сцен)

Предмет: исследование абстрактных моделей графических объектов и взаимосвязей между ними. Объекты могут быть как синтезированными, так и выделенными на фотоснимках.

1 шаг в анализе сцены — выделение характерных особенностей, формирующих графический объект(ы).

Примеры: машинное зрение (роботы), анализ рентгеновских снимков с выделением и отслеживанием интересующего объекта, например, сердца.

Итак, в основе перцептивной графики лежат:
изобразительная графика + анализ изображений + специализированные средства.

Когнитивная графика
(только формирующееся направление)

База: мощные ЭВМ и высокопроизводительные средства визуализации.

МГ для научных абстракций, способствующая рождению нового научного знания.

Последовательность познания

Гипотеза — Модель — Решение.

2 механизма мышления:

- **логическое, манипулирует абстрактными последовательностями символов + семантика символов + прагматические представления, связанные с символами;**
- **образное, интуитивное, работает с чувственными образами и представлениями о них.**

Этапы использования ЭВМ

1. **Малые производительности процессоров и средств МГ — работа только с символами (некий, упрощенный вариант /аналог/ логического мышления).**
2. **Супер ЭВМ (Гига- и терафлопы) и супер станции — средства работы с картинами (образами).**

Мозг не только манипулирует двумя представлениями, но и соотносит их, совершая переходы логическое $\leftrightarrow$ образное.

Проблема когнитивной МГ — создание таких моделей представления знаний, в которых однообразно представимы объекты, характерные как для логического (символического) мышления, так и объекты характерные для образного (интуитивного, геометрического) мышления.

Важнейшие задачи когнитивной МГ

1. Визуализация таких знаний, для которых не существует (пока) описания в символах.
2. Поиск путей перехода от образа к формулировке гипотезы о механизмах и процессах, представленных этими динамическими образами на экране дисплея.

Появление когнитивной МГ — переход от эры экстенсивного развития познания к эре интенсивного развития, к технологии человеко-машинного познания с непосредственным, целенаправленным, активирующим воздействием на подсознательные, интуитивные механизмы образного мышления.

Один из самых ранних и ярких примеров: Ч.Страус Неожиданное применение ЭВМ в чистой математике. ТИИЭР, т. 62, ь4, 1974, с. 96-99.

В статье показано как в “чистой” математике для анализа окрестностей особых точек алгебраических кривых в комплексном пространстве используется “n–мерная” доска на основе графического терминала, с использованием средств ввода для направленного изменения параметров.

В процессе взаимодействия углубляется понимание роли вариаций параметров.

В результате получено “несколько новых теорем и определены направления дальнейших исследований”.

Приложения МГ

- **Компьютерное моделирование**
- **CAD/CAM, САНИ, АСУТП**
- **Бизес**
- **Искусство**
- **Средства массовой информации**
- **Досуг**
- **Виртуальная реальность**
- **. . .**

Цели компьютерного моделирования

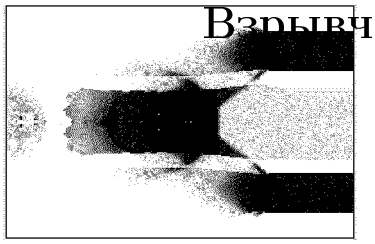
1. **Исследование процессов и явлений в областях где натурный эксперимент затруднен или невозможен:**

- **разработка систем оружия,**
- **экологические катастрофы,**
- **сварка, упрочнение и прессование взрывом,**
- **получение композитов и искусственных алмазов,**
- **добыча полезных ископаемых,**
- **противометеоритная защита,**
- **генерация сверхмощных магнитных полей,**
- **взрывные источники тока,**
- **защита экологически опасных объектов.**

Э.Теллер в Челябинске-70 возле самой крупной Н-бомбы мира

2. **Унификация понимания и представлений**

В качестве иллюстрации некоторых явлений, которые необходимо моделировать, ниже показан один из этапов решения осесимметричной задачи обжатия капсулы с газом с помощью взрыва.



Обжатие капсулы с газом.

Этапы разработки

Характеристики задач моделирования быстропротекающих процессов

Параметры	Вид задачи			
	Тестовая	Малая	Средняя	Большая
Расчетное поле, ячеек	256x128	256x128	512x128	1024x256
Рассчитываемых параметров	5–15	5	15	15
Объем данных на шаге, Кбайт	640–1920	640	3840	15360
Шагов по времени	$\sim$100	2048	2048	2048
Суммарный объем данных, Мбайт	63–188	1280	7680	30720
Шаг вывода кадров изображений	1	10	10	10
Объем кадра (8 бит/пиксел), Кбайт	32	32	64	256
Суммарный объем кадров для одного параметра, Мбайт	3.1	6.4	12.8	51.2

Проблемы компьютерного моделирования

гигантские объемы данных — десятки Гбайт
2D задача физики взрыва, 2048 шагов,
1024×256 ячеек, 15 параметров на ячейку,
время расчета ∼ 1.7 часа (400 млн SIMD-машина)
формат вывода чисел ±.XXXXXXX±XX
печать листами A4, 40 строк по 62 символа
число листов ∼ 38 966 437
бумага бы стоила ∼ 2 млрд 338 млн р (цены 1997)
время печати на офисном HP LJ ∼ 9.3 года
высота стопки ∼ 3 897 м
длина стопки ∼ 11 573 км

ВЫЧИСЛЯЮТ БОЛЬШЕ ЧЕМ МОГУТ ЗАПОМНИТЬ.
ЗАПОМИНАЮТ БОЛЬШЕ ЧЕМ МОГУТ ПОНЯТЬ.

Один из путей решения — использование графики.

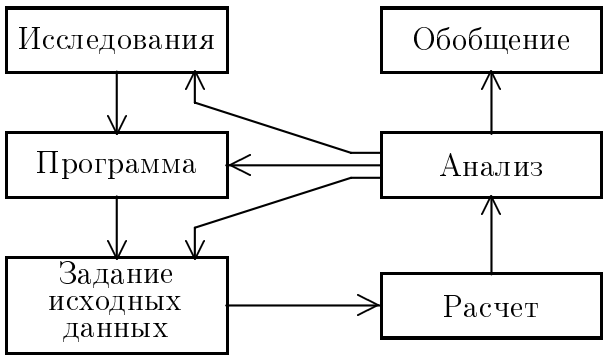
Одномерные задачи — тонут в цифрах.
Адекватное решение — вывод графиков.

Двумерные задачи — тонут в графиках.
Адекватное решение — машинные фильмы + пересчет.

Трехмерные задачи — тонут в машинных фильмах.
Возможное решение — иные методы вычислений.

Технология решения и требования к средствам машинной графики

- **диалоговые постановка задачи и задание исходных данных;**
- **счет с оперативным отображением и накоплением результатов;**
- **диалоговые формирование и просмотр машинных фильмов;**
- **анализ и обобщение результатов расчетов.**



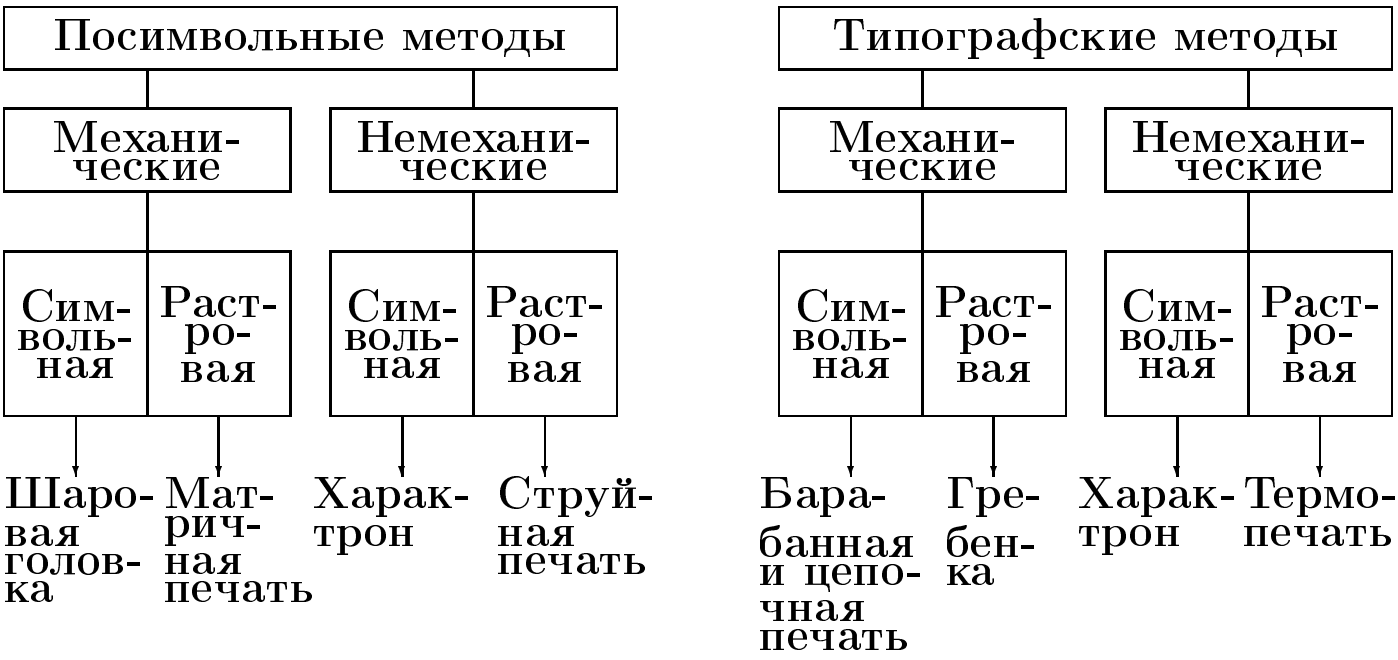
Технические средства

1. **Классификация методов печати**
   - **механические;**
   - **немеханические;**
   - **посимвольные и типографские.**
2. **Печатающие устройства**
   - **матричный принтер;**
   - **струйная печать;**
   - **лазерный принтер.**
3. **Графопостроители**
   - **планшетные;**
   - **с перемещающимся носителем;**
   - **электростатические.**

Механические методы

Немеханические методы

Посимвольные и типографские методы

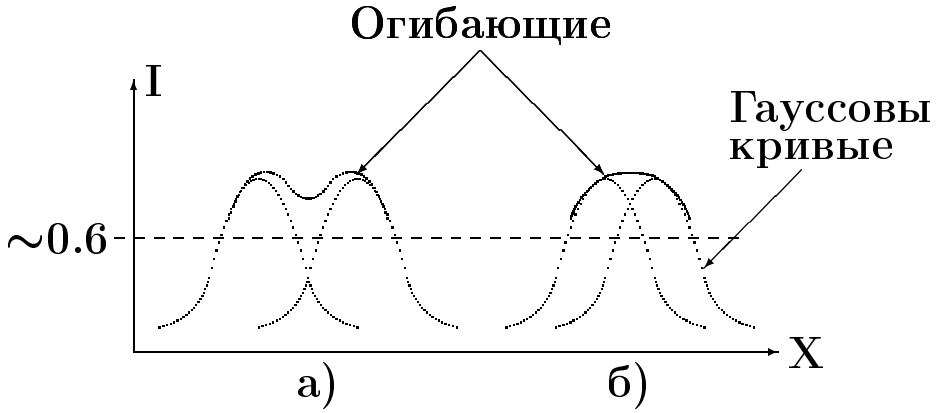


Разрешение устройств печати

Пространственное разрешение растровых устройств определяется:

- размером элементарного пятна,
- формой пятна (круглое, эллипсоидальное или иное),
- распределением яркости по пятну (по гауссу или иное),
- возможностями размещения отдельных пятен (адресация и фактическое размещение).

Физическое разрешение растровой печати



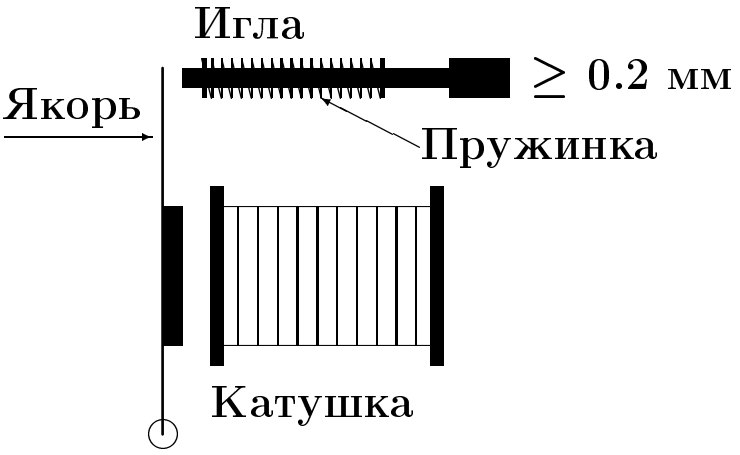
Если пересечение пятен ниже уровня ≈0.6 от максимальной интенсивности, то пятна различимы (рис. а).

Если же пересечение пятен выше уровня 0.6, то пятна неразличимы (рис. б).

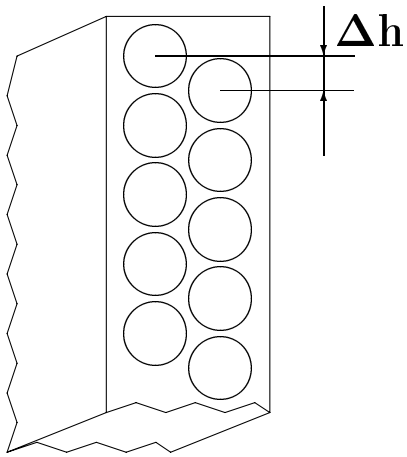
Матричный принтер

- черно-белые,
- цветные.

- $7 \div 48$ вертикальных штырьков (игл),
- основные — 9 и 24-х игольчатые



а)



б)

Разрешение

- 85–127 точек/дюйм (3–5 точек/мм);
- для 48-игольчатых принтеров $\approx$ 300 точек/дюйм;
- адресное разрешение по оси Y определяется расстоянием между иглами (Δh на рис. б) и составляет до 170 точек/дюйм.

Красящая лента

- ширина $1/4 \div 2$ дюйма ($\approx 6 \div 50$ мм);
- основные ленты $\sim 1/2$ дюйма (≈ 13 мм);
- для продления срока службы ленты используются или кольцо Мебиуса или вертикальные смещения ленты относительно головки;

Цветные принтеры

- одно- и многопроходные;
- медленные — смещение ленты по вертикали;
- быстрые — несколько проходов по бумаге.

Основные свойства матричных принтеров:

- неналожение точек различных цветов друг на друга — забота пользователя;
- несколько комплектов внутренних шрифтов;
- память для загрузки пользовательских шрифтов;
- оба хода головки рабочие и слева-направо и справа-налево;
- встроенная память для накопления распечатываемого текста (до нескольких десятков страниц).

Режимы работы:

- символьный — принтер сам управляет печатью строк, используя внутреннее растровое описание шрифтов;
- графический – пользователь должен подготовить поточечное описание строк.

Скорости печати:

- символов — 120, 120–200, $>$ 200 символов/мин;
- растровых строк — до 200, 200–400, более 400 строк/мин.

Обозначение наивысшего качества печати:

- NLQ — Near Letter Quality — качество, близкое к качеству пишущей машинки (обычно 9-ти игольчатые);
- LQ — Letter Quality — качество пишущей машинки (обычно 24-х игольчатые).

Стандарты де-факто на протоколы матричного принтера:

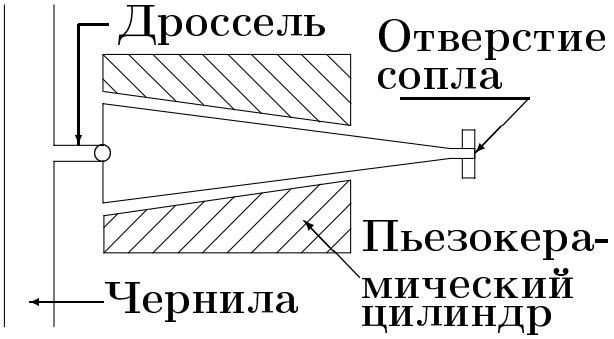
- система команд фирмы Epson;
- система команд фирмы IBM для семейства принтеров Proprinter;
- система команд фирмы IBM для графического принтера (IBM Graphics printer).

Струйные принтеры

Основные характеристики:

- невысокая цена,
- хорошее разрешение,
- практически бесшумная работа,
- небольшое энергопотребление,
- относительно невысокая скорость.

Устройство отдельного сопла струйного принтера



- сопла одного цвета вертикально размещается в печатающей головке;
- частота работы сопел — до 900 Гц;
- для цветной печати обычно используется три цвета — желтый, голубой, малиновый. Часто добавляется черный цвет.

Разрешающая способность:

	Горизонтальная	Вертикальная
Обычная	150 dpi (6 точек/мм)	100 dpi (4 точки/мм)
Современная	$>$ 300 dpi	300 dpi (12 точек/мм)
	до 50 сопел на 1/6 дюйма	

Скорость печати:

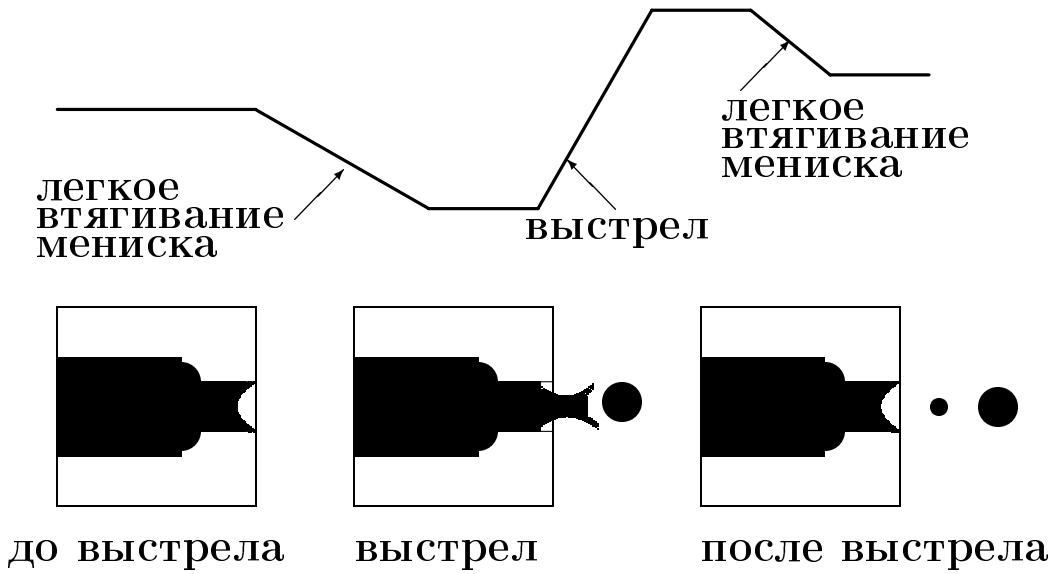
- $20 \div 50$ символов/с;
- $\approx$ 90 с на лист 210×297 мм в графическом режиме.

Струйный принтер фирмы HP

- выбрасывание капель под воздействием теплового удара;
- самая низкая цена;
- невысокое разрешение.

Струйный принтер фирмы Seiko Epson Corporation

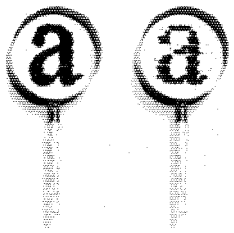
Новая технология струйной печати (Micro Piezo, Micro Dot и Micro Wave), основанная на управлении мениском чернил в сопле.



Характеристики принтера с технологией Micro Dot:

- управление размером и формой чернильных пятен,
- повышение скорости выстреливания капель,
- увеличение количества оттенков до шести, включая полутона,
- устранение зернистости,
- повышение разрешения до 1440 dpi ($\approx$ 57 точек/мм).

Струйная печать по технологиям Micro Dot и традиционной

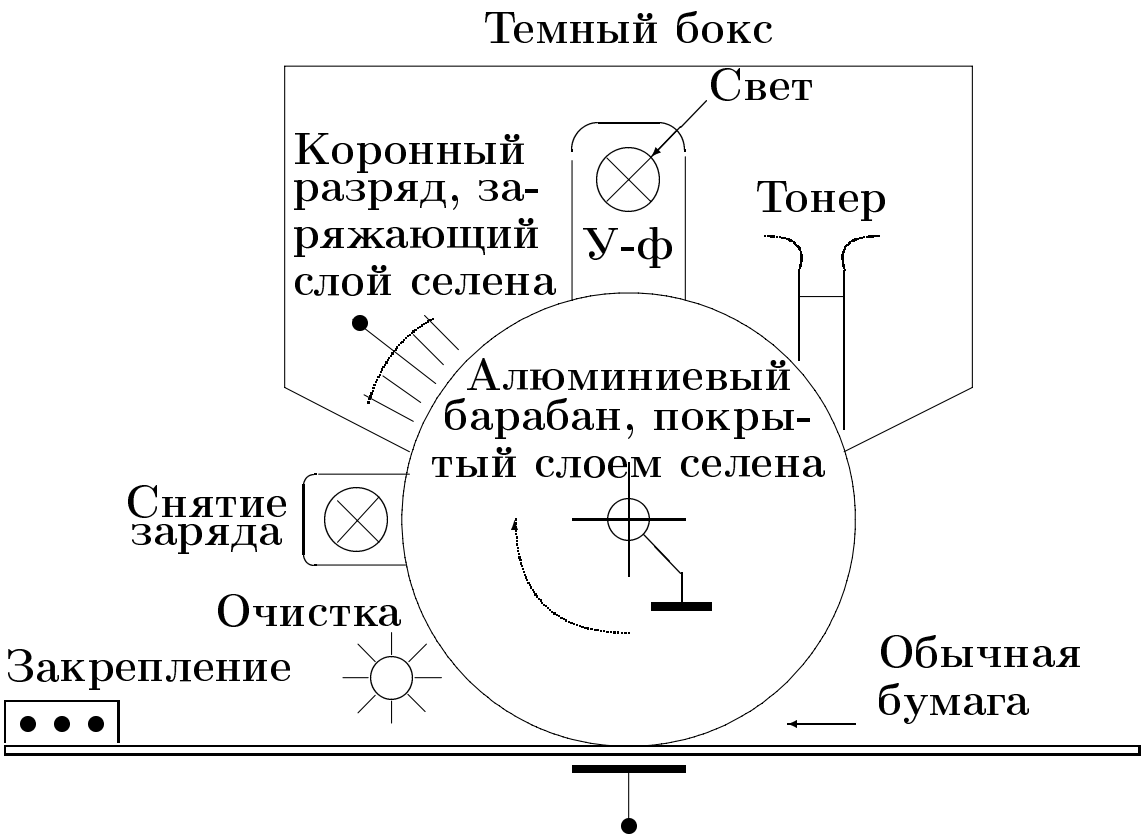


Протокол устройств струйной печати

- стандарт де-факто — протокол принтеров фирмы Epson,
- ряд принтеров эмулирует язык HPGL, разработанный фирмой Хьюлетт-Паккрад для графопостроителей.

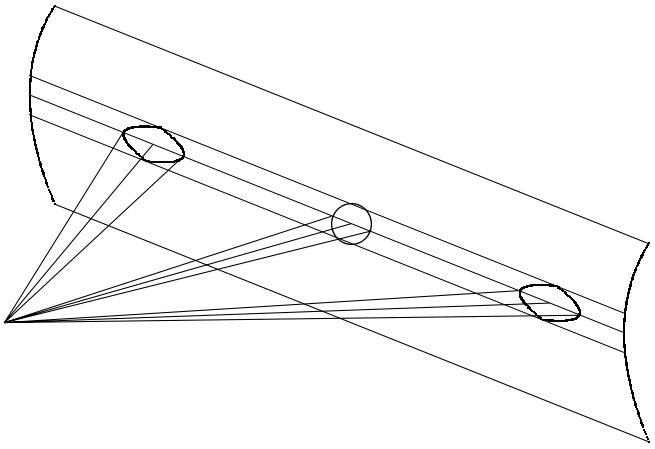
Лазерный принтер

- метод печати — ксерографический (электрофотографический)



Разрешающая способность:

- типовая — 600 dpi (23.6 точек/мм),
- достижимая — 1200 dpi (47.2 точек/мм).



Органичение в разрешении лазерных принтеров с отклоняющимся лучом связано с различной формой пятна в центре барабана (круг) и на краях (эллипс).

Скорости

- настольные офисные — 4–8 страниц/мин,
- стационарные, с двухсторонней печатью — порядка 150 страниц/мин.

Протоколы лазерных принтеров

- стандарт де-факто — система команд PCL (язык команд принтера) лазерных принтеров HP LaserJet фирмы Hewlett Paccard.
  Описание шрифта — битовые матрицы символов, поэтому для разных масштабов нужны версии одного и того же шрифта.
- широко распространены языки описания страниц (PDL — Page Description Language).
  Стандарт де-факто — язык описания страниц PostScript. Другие распространенные PDL:
  — Interpress фирмы Xerox,
  — DDL фирмы Imagen.
  Описание шрифта в PostScript — схема его символов, изменяемая программным путем, поэтому для разных масштабов не требуются версии шрифта.

Описание картинки на языке PostScript обычно велико:
$<$ 10 Кбайт на PCL $\Longrightarrow >$ 100 Кбайт на PostScript.

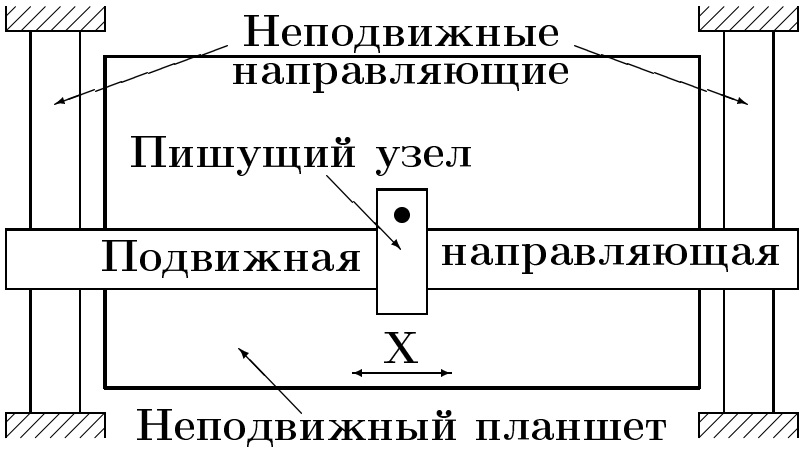
Графопостроители

Назначение — высококачественное документирование чертежно-графической информации.

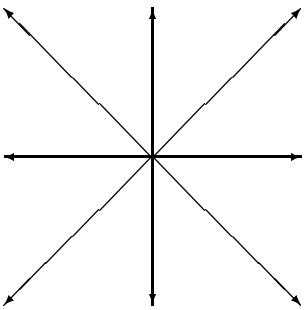
Классификация:

- по способу формирования чертежа — с произвольным сканированием и растровые;
- по способу перемещения носителя — планшетные, барабанные и смешанные (фрикционные);
- по используемому инструменту (типу чертежной головки) — перьевые, фотопостроители, со скрайбирующей или с фрезерной головкой.

Планшетные графопостроители с подвижным инструментом



- размер чертежа;
- требования к носителю чертежа;
- закрепление носителя.

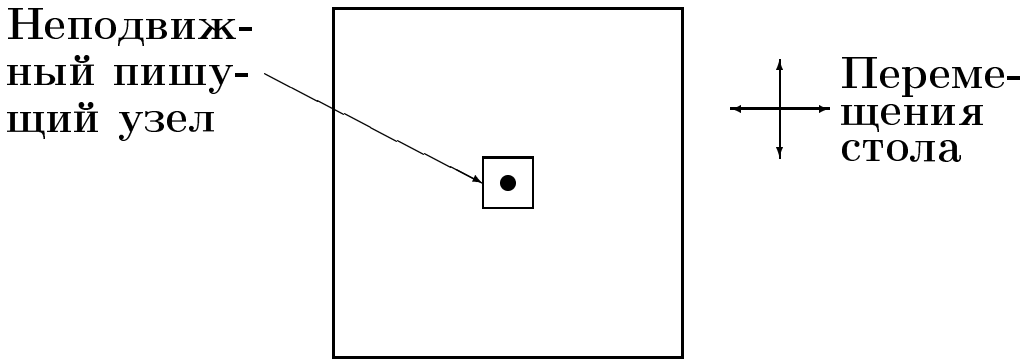


Приводы графопостроителей:

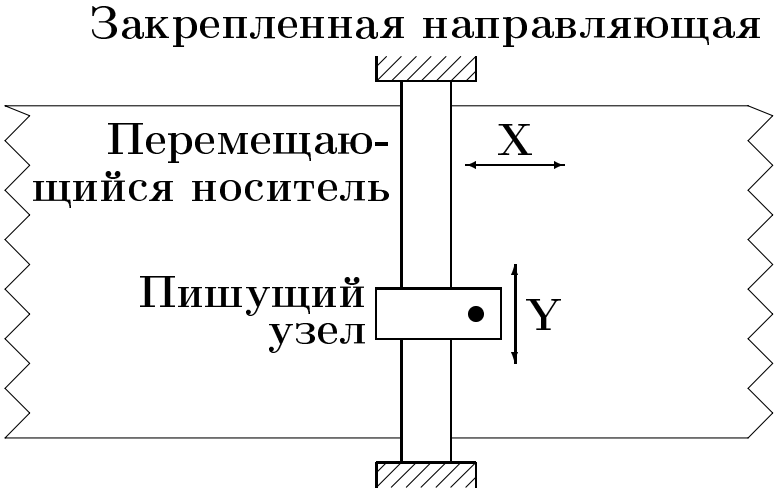
- на шаговых двигателях;
- исполнительная система с обратной связью.

Проблемы аппроксимации произвольных линий отрезками 8 направлений.

Планшетные графопостроители с подвижным столом



Графопостроители с перемещающимся носителем

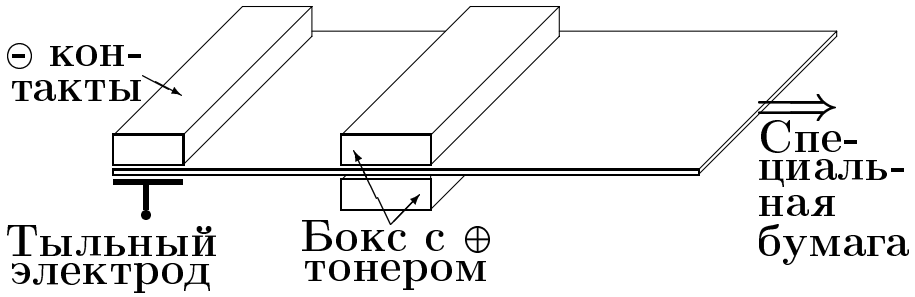


Разновидности:

- барабанные графопостроители
  (размер носителя фиксирован);
- рулонные графопостроители
  (специальный носитель с краевой перфорацией);
- фрикционные графопостроители
  (обычный носитель перемещается с помощью фрикционных роликов,
  новый вариант — графопостроитель с т.н. абразивной головкой).

Графопостроители с растровым принципом формирования изображений

Цветные электростатические графопостроители



⊖ заряженные иголки до 16 шт/мм
⊕ заряженный тонер

Разрешение	200÷400 точек/дюйм (8÷16 точек/мм)
Скорость	500÷1000 линий/мин
	(в 10÷20 раз быстрее перьевого)
Стоимость	\$30 000÷\$150 000
Фирмы	Versatec, Calcomp, Benson

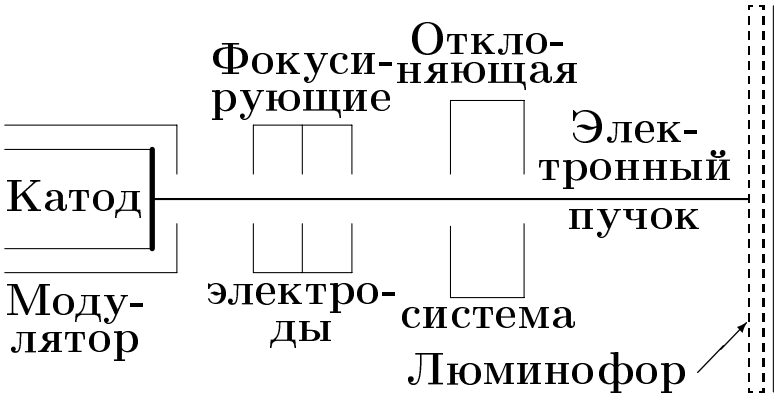
Основные параметры графопостроителей

Параметр	Планшетные		Барабанные	
Формат	Ширина	Длина	Ширина	Длина
черчения, мм	210–840	297–1188	210–1140	297–не ограничена
Скорость черчения	80 – 1140 мм/с		30–300 мм/с	
Точность	0.8 – 0.0025 мм		0.7–0.0025 мм	
Разрешение	0.4 – 0.0025 мм		0.1–0.0025 мм	

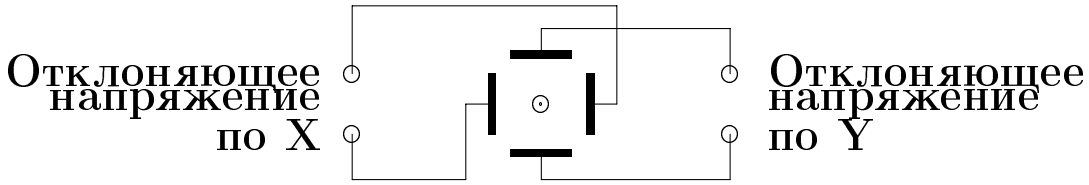
Протоколы графопостроителей

- графопостроители общего (широкого) применения:
  HPGL — Hewlett Packard Graphics Language,
  REGIS — (на некторых графопостроителях);
- уникальные графопостроители.

Черно-белые кинескопы



а) Схема устройства ЭЛТ



б) Электростатическая отклоняющая система

U $\approx$ 10 Кв, I $\approx$ 10 мка,
диаметр пятна $\approx$ 0.25 мм
P $\approx$ 150 Вт/см^2 (электроплитка $\approx$ 7 Вт/см^2)

Векторные и растровые дисплеи

- классификация устройств вывода
- экскурс в историю
- общая архитектура графической системы
- дисплейные файлы
- векторные дисплеи
- растровые дисплеи

Классификация устройств вывода

1. По принципам записи (обновления) изображения:
- каллиграфические устройства (штриховые, с произвольным сканированием луча);
- сканирующие устройства (растровые).

2. По принципам отображения:
- периодическая регенерация информации на экране из неотображающей памяти;
- использование отображающего устройства сохранения изображения.

3. По технологическим способам вывода.

Векторные дисплеи

- изображение строится из отдельных отрезков;
- регенерация картины перевыдачей из неотображающей памяти;
- вывод — свечение люминофора.

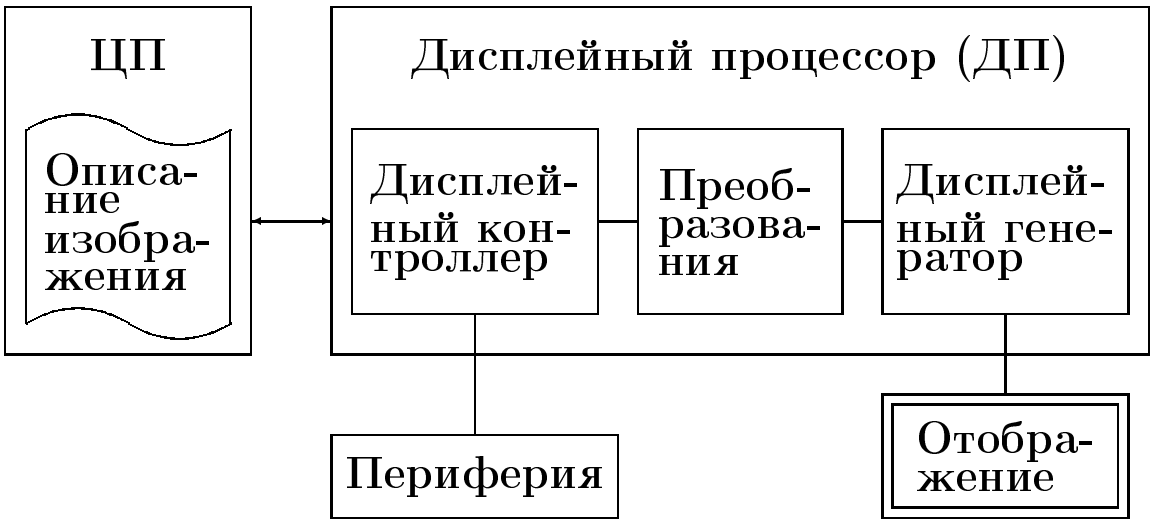
Растровые дисплеи

- изображение и вывод отделены;
- регенерация картины построчным сканированием неотображающей видеопамяти, содержащей массив пикселов (picture element);
- вывод — свечение люминофора.

История дисплеев

- первые серийные (векторные) дисплеи за рубежом — конец 60-х.
  Аналоговая схемотехника отклонения луча.
  1963 — прототип IBM 2250. До осени 1964 работы были засекречены.
- 1972, ИАиЭ, векторный дисплей Символ.
- 1973, ИАиЭ, векторный дисплей Дельта. Малая серия 1974 ИПФ (Новосибирск), серия — 1975, завод Луч (НЭМЗ) ММ СССР, 1975.
- 1977, ИАиЭ, векторный дисплей ЭПГ-400.
- 1981–1982, векторные дисплеи ЭПГ-СМ (ИАиЭ), ЭПГ-2СМ.
- 1982, Киев, НИИПО, СМ-7316 (4000–6000 векторов, 4096 символов, разрешение 2048×2048).
- 1969, ВЦ АН СССР, исторически первый растровый дисплей. видеопамять на МБ весом 400 кГ.
- 1979, ИПФ, растровый дисплей Гамма-1.
- 1982, ИПФ, растровый дисплей Гамма-2.
- 1983, ИПФ, растровый дисплей Гамма-4, серия.
- 1984, ИПФ, растровый дисплей Гамма-5, серия.

Общая архитектура графической системы



Описание изображения (дисплейный файл)

- линейный дисплейный файл,
- сегментированный дисплейный файл,
- структурированный дисплейный файл.

Линейный дисплейный файл

Набор команд описания изображения, зачастую непосредственно команд дисплейного процессора:

- позиционирования луча,
- построения отрезка,
- построения символов,
- задания геометрических атрибутов построения,
- задания цветовых атрибутов построения,
- команды управления (конец файла),
- возможны команды передачи управления,
- возможны команды организации циклов.

Сегментированный дисплейный файл

Линейный дисплейный файл, позволяющий независимую модификацию частей изображения. Расширяется набор команд управления. Обязательно имеются команды передачи управления. Возможно имеются команды вызова подпрограмм изображения.

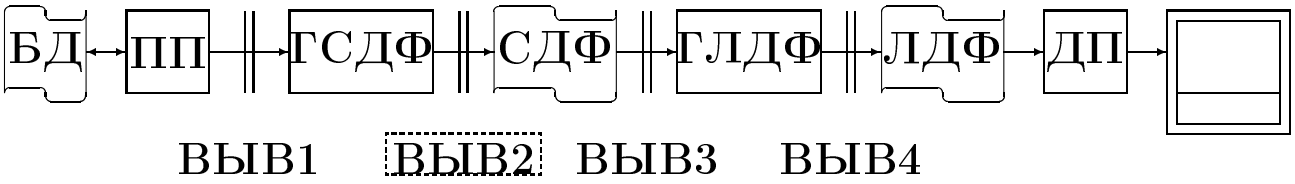
Структурированный дисплейный файл

Предусматривает описание изображения в виде дерева в общем случае вложенных вызовов подпрограмм.

Резко сокращается объем передаваемых данных при модификации картины.

- обязательны команды вызова подпрограммы;
- обычно имеются команды организации циклов;
- желательны команды выполнения преобразований изображения и ведения стека преобразований.

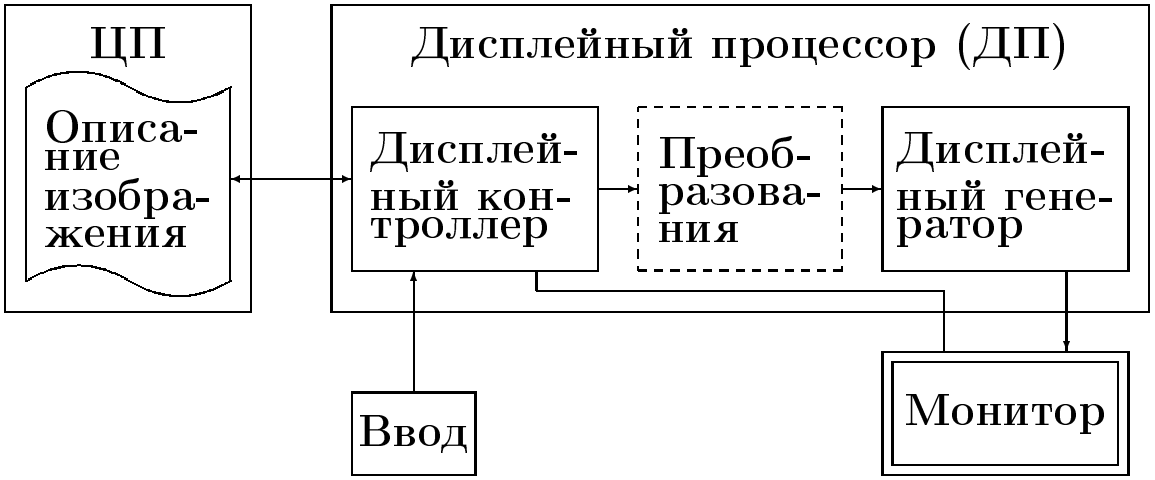
Модель процесса вывода



Наиболее целесообразные границы раздела между процессорами ВЫВ1, ВЫВ3 и ВЫВ4.

ВЫВ1 и ВЫВ3 обеспечивают аппаратную независимость, а ВЫВ4 соотвествует минимальным требованиям к подсистеме отображения.

Векторные дисплеи



Дисплейный генератор:

- векторов (ЦДА, Брезенхема);
- символов (векторный, матрицы точек);
- кривых (поверхности, дуги окружностей, эллипсов);
- двойная буферизация.

Преобразования:

- координатные;
- отсечение.

Дисплейный контроллер:

- связь с компьютером;
- управление формированием изображения;
- обработка графического ввода.

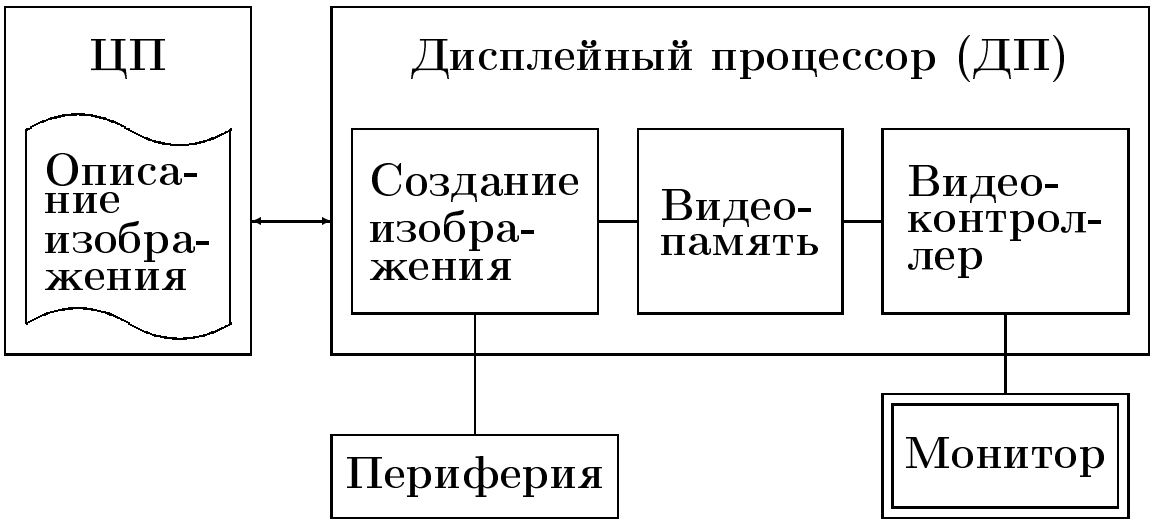
Растровые дисплеи

- изображение состоит из строк пикселов (picture element);
- процесс вывода на экран (50–80 Гц) не зависит от построения;
- сложность немерцающего изображения не ограничена.

Преимущества:

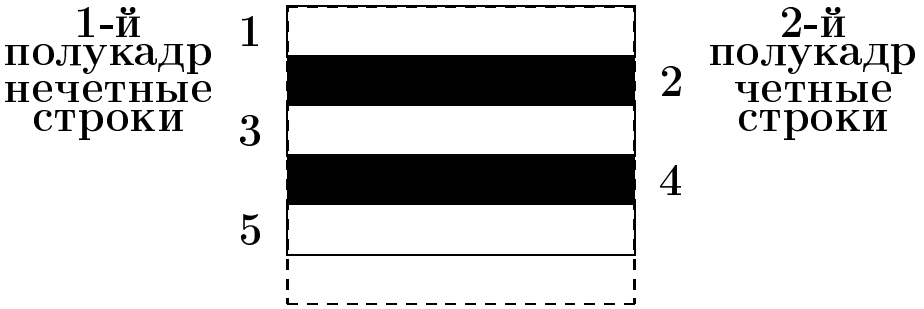
- наивысшее качество при меньшей стоимости,
- смешение синтезированного и телевизионного изображений,
- интерактивная машинная графика и обработка изображений могут выполняться в рамках одной системы,
- телевизор хорошо знаком каждому.

Компоненты растрового дисплея

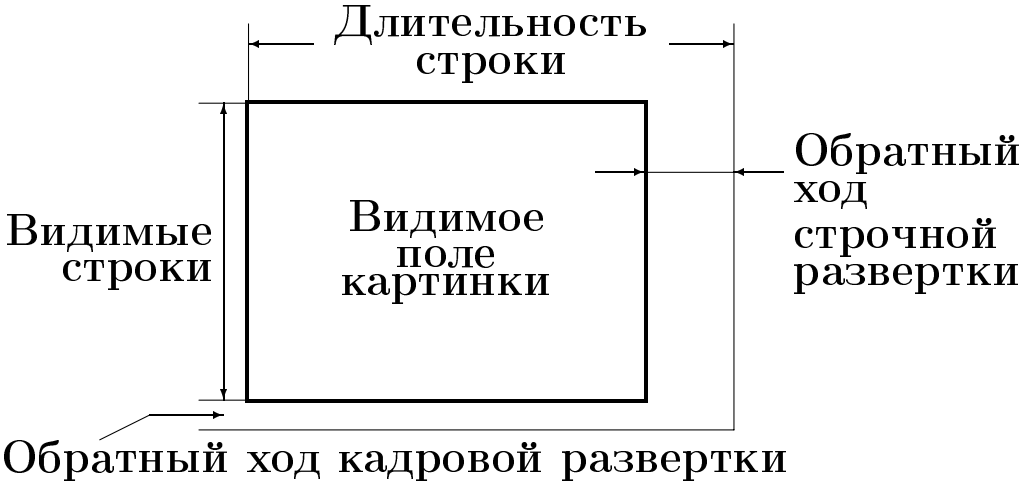


Видеоконтроллер

- формирует изображение на экране монитора,
- построчная и чересстрочная (Interlacing) развертки.



Развертки



Время, требуемое на обработку строки:

$$t_s = \frac{1/ \ - t}{}$$

Время, требуемое на обработку одного пиксела:

$$t_n = \frac{t_s - t}{}$$

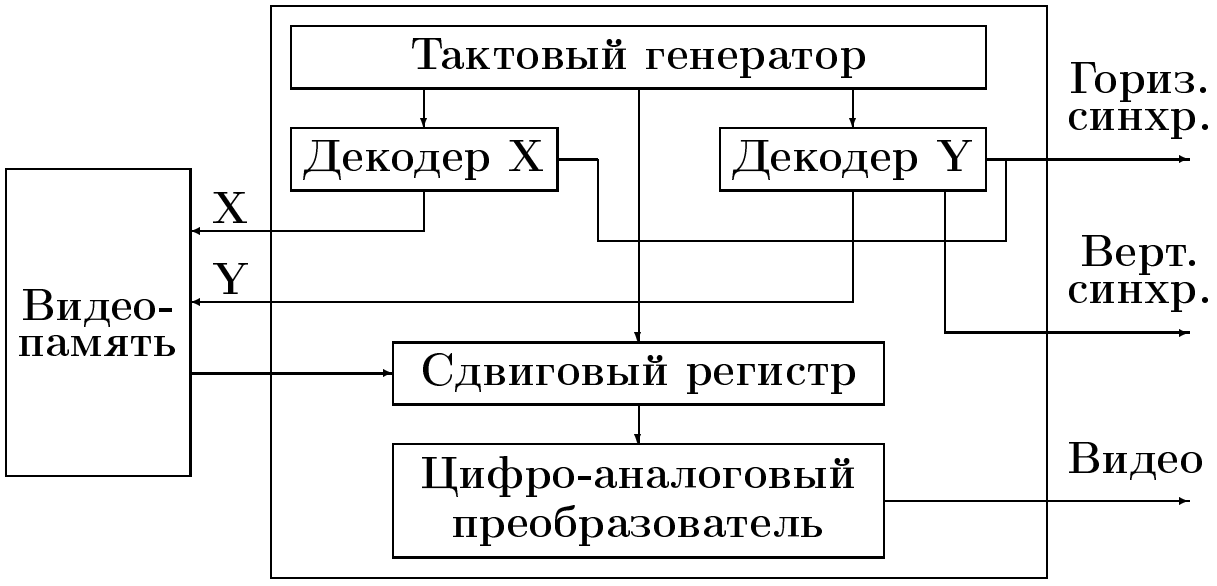
Характерные времена для некоторых мониторов

Разрешение X×Y	f кадр. Гц	t полн. строки мкс	t обр. строки мкс	t обр. кадров мкс	Чересстрочность	tп на пиксел нс
512×512	30	60.4	10.9	1203	да	96.7
768×576	50	64	12	1612	да	70.3
1024×768	60	20.92	4	600	нет	16.52
1024×1024	60	15.69	4	600	нет	11.42
1280×1024	60	15.69	4	600	нет	9.13

Видеоконтроллер

- адресация и чтение данных из видеопамяти;
- формирование синхроимпульсов разверток по горизонтали и вертикали, соответствующих формату изображения. Эти синхроимпульсы используются монитором для формирования отклоняющих напряжений;
- управление монитором для задания требуемых цветов и интенсивностей, цифро-аналоговое преобразование.

Простой видеоконтроллер

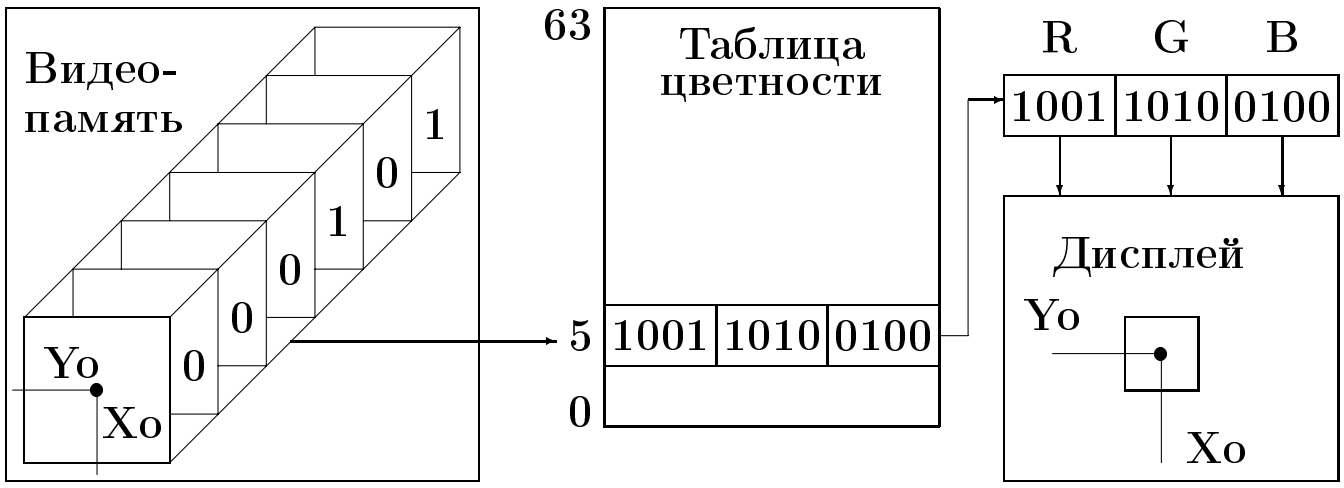


Типы растровых дисплеев

- полноцветные дисплеи, в которых для каждого пиксела сразу хранятся значения R, G, B.
- дисплеи с таблицей цветности (ТЦ, Look Up Table — LUT), в которых значение считанного пиксела используется как адрес в таблице цветности. По этому адресу выбираются значения яркостей по R, G, B и уже они передаются на ЦАП;
- смешанные типы.

Отличительные свойства дисплеев с ТЦ:

- экономия видеопамяти;
- перекраска изображения в темпе разверток без модификации данных в видеопамяти;
- простая реализация различных динамических эффектов;
- простая реализация фильтрации;
- простая реализация гамма-коррекции.



Управление плоскостями изображения

6-ти битовый дисплей (64 элемента в ТЦ).

Включение зеленым цветом изображения в плоск. 0

Код пиксела		Значение
дес.	двоичн.	элем. ТЦ
0	000000	0 0 0
1	000001	0 G 0
2	000010	0 0 0
3	000011	0 G 0
4	000100	0 0 0
	...	...
62	111110	0 0 0
63	111111	0 G 0

Включение красным цветом изображения в плоск. 1

Код пиксела		Значение
дес.	двоичн.	элем. ТЦ
0	000000	0 0 0
1	000001	0 0 0
2	000010	R 0 0
3	000011	R 0 0
4	000100	0 0 0
	...	...
62	111110	R 0 0
63	111111	R 0 0

Эффекты перемещений с помощью ТЦ

- генерация картинки нескольких изображений одного и того же тела, например, шарика, каждое своим кодом пиксела. Перекрашивая ТЦ создаем эффект движения;
- генерация изображения, например, кривой с изменением кодов пикселов вдоль кривой. Меняя раскраску ТЦ может создать впечатление переливов цвета вдоль кривой.

Приоритет изображений

3-битовый дисплей. Три картинки А, Б и В. Старшая плоскость — изображение А с max приоритетом. Младшая плоскость — изображение В с наименьшим приоритетом.

Значен. пиксела	Содержим. ТЦ при перекрытии слоев	Содержим. ТЦ при прозрачных слоях
0 0 0	Цвет фона	черный
0 0 1	Цвет картинки В	синий
0 1 0	Цвет картинки Б	зеленый
0 1 1	Цвет картинки Б	изумрудный
1 0 0	Цвет картинки А	красный
1 0 1	Цвет картинки А	пурпурный
1 1 0	Цвет картинки А	коричневый
1 1 1	Цвет картинки А	белый
Картина: А Б В	Приоритет А> Б> В	Прозрач. слои

Преобразование изображений “на лету”

Строка	Адрес
0	0000
1	1024
. . .	
448	458752
449	459776
. . .	
1021	1045504
1022	1046528
1023	1047552

Видеопамять 1024×1024

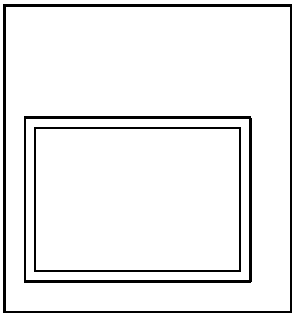
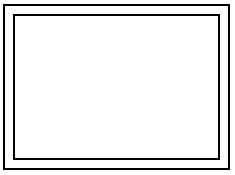


Таблица адресов

Строка	Адрес	Масштаб
0	1047562	1:1
1	1046538	1:1
. . .		
573	460810	1:1
574	459786	1:1
575	458762	1:1

Экран 768×576



- формирование и вывод разделены — сложность практически не ограничена, указание затруднено;
- динамическое представление “не дергающихся” изображений требует совпадения скоростей генерации и видеовывода.

Операции при занесении

Набор команд для растровых устройств, кроме вывода векторных примитивов, обычно включает:

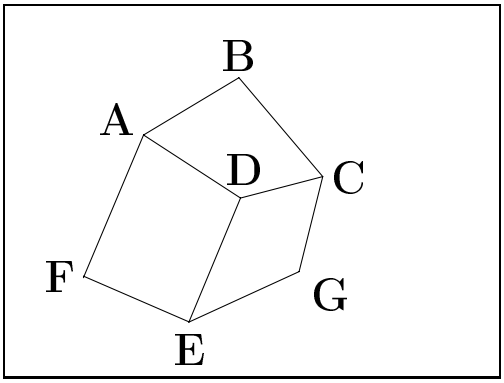
- задание яркостей, цветов, закраска поверхностей;
- команды копирования блоков пикселов Bitblt (Bit Boundary Block Transfer) или RasterOp (растровые операции);
- выполнение логических операций при занесении.

Практически используются следующие операции:

Очистка	приемник:= константа
Узор	приемник:= постоянный узор
Копирование	приемник:= источник
Инвертирование	приемник:= NOT приемник
Логическое И	приемник:= источник AND приемник
Логическое ИЛИ	приемник:= источник OR приемник
Исключающее ИЛИ	приемник:= источник XOR приемник

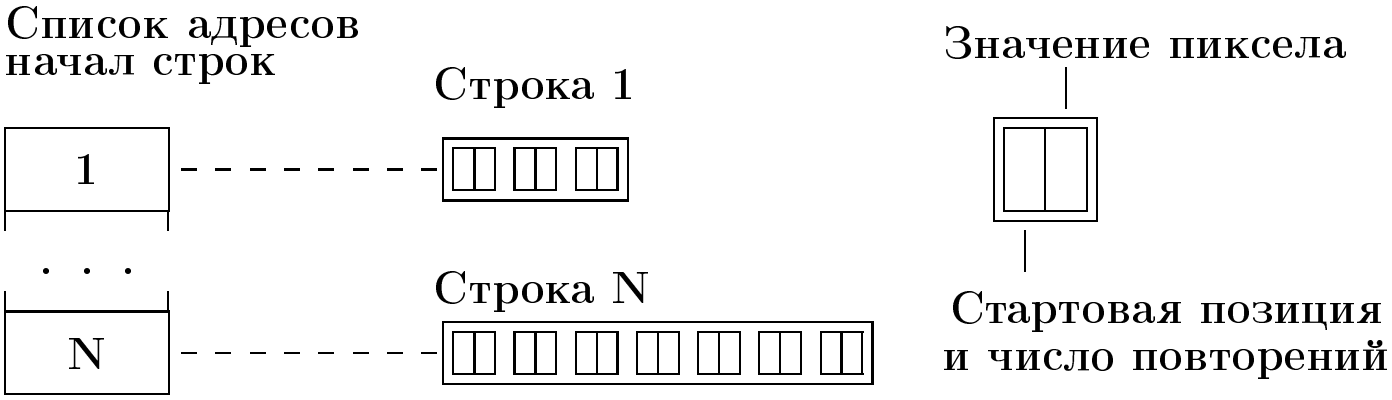
Особенно интересна операция XOR(eXclusive OR):
(Приемник XOR Источник) XOR Источник =
Приемник XOR (Источник XOR Источник) =
Приемник XOR 0 = Приемник

Раскраска изображения "на лету"



Кодирование данных в видеопамяти

- кодирование длин серий (*.PCX);
- клеточное кодирование (символы).



Повышение быстродействия подсистемы визуализации

1. Растрирование в реальном времени.
2. Распараллеливание.

Растрирование в реальном времени — без сохранения данных в видеопамяти. При отображении пиксела на экран требуется обработка всей сцены за 16.5 нс для дисплея 1024×768! При использовании буфера строки на обработку описания сцены требуется 20.9 мкс).

Распараллеливание.

- распараллеливание последовательных алгоритмов;
- процессор (параллельный) на графический элемент;
- процессор на пиксел.

Распараллеливание алгоритма Брезенхема на N процессоров для отрезка:
$(X_0Y_0,\ X_1Y_1)$ с $\Delta X = X_1 - X_0\ >\ \Delta Y = Y_1 - Y_0$

$$\begin{aligned} \Delta X_p &= \frac{\Delta X + N - 1}{N} \\ \Delta Y_p &= \frac{\Delta Y}{\Delta X} \cdot \Delta X_p \end{aligned}$$

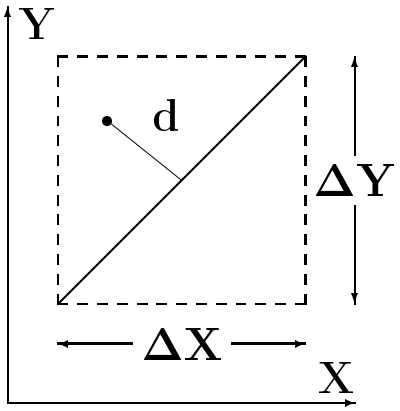
для k-го процессора

$$\begin{aligned} X_{k,0} &= X_0 + k \cdot \Delta X_p, \\ Y_{k,0} &= Y_0 + \mathrm{round}(k \cdot \Delta Y_p), \\ E_{k,0} &= (k\Delta X_p) \cdot (2\Delta Y) - \mathrm{round}(k \cdot \Delta Y_p) + 2\Delta Y - \Delta X \end{aligned}$$

Процессор на элемент, процессор на пиксел

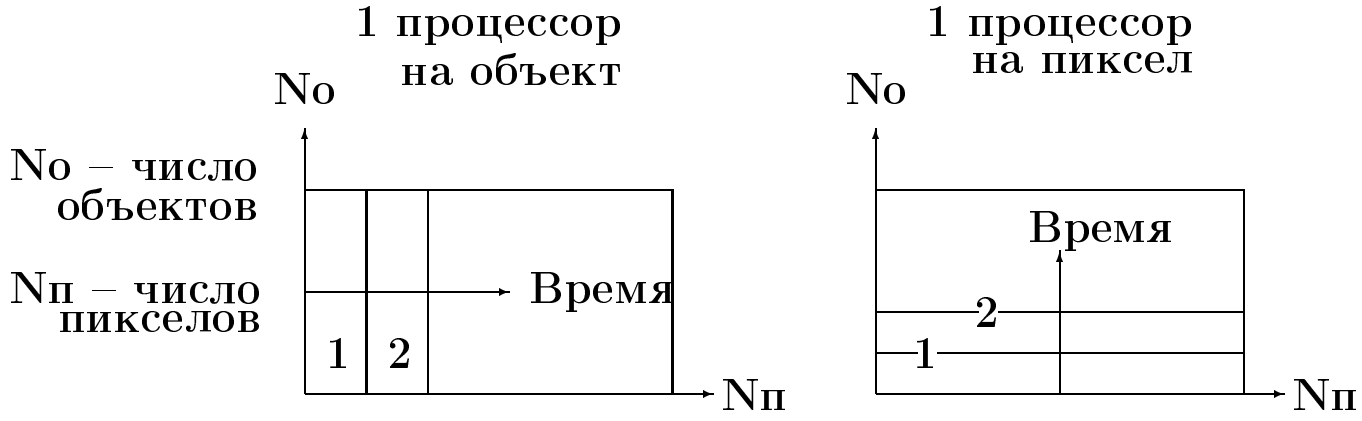
Процессор на графический элемент — генераторы для отрезков, кривых, поверхностей и т.д.

Процессор на пиксел (группу пикселов) на примере генерации отрезка.



$$
\begin{aligned}
d &= Ax + By + C, \\
A &= -\Delta Y/len, \\
B &= \Delta X/len, \\
C &= (X_0\Delta Y - Y_0\Delta X)/len, \\
len &= \sqrt{\Delta X^2 + \Delta Y^2}.
\end{aligned}
$$

После вычисления d для каждого пиксела в области $(\Delta X, \Delta Y)$ сравнением с требуемой толщиной линии определяется подсвечивать его или нет.



Видеопамять

Основные функции:

- занесение и сохранение сгенерированного изображения;
- обеспечение вывода изображения.

Ширина полосы пропускания видеопямяти:

- максимальный объем записываемых и читаемых данных в видеопамять в секунду.

Проблемы видеопамяти:

- конкуренция по доступу от генератора и видеовывода,
- обеспечение требуемой скорости работы.

Решение проблем конкуренции по доступу:

- не решать — по запросу генератора ему отдаются циклы, отбираемые у видеовывода;
- генератору дается время, свободное от видеовывода (обратный ход строчной и кадровой развертки);
- использование буфера на строку — строка видеопамяти копируется в буфер строки. Видеовывод идет из буфера строки. Видеопамять отдается генератору.
- использование памяти специальной, двухпортовой организации, например, VRAM.

Обычная организация видеопамяти — послойная. Нужная глубина пиксела обеспечивается комплексированием слоев.

Как правило используется динамическая память с произвольным доступом (DRAM — Dinamic Random Access Memory), обычно имеющая организацию N×1 разряд (однобитовая память).

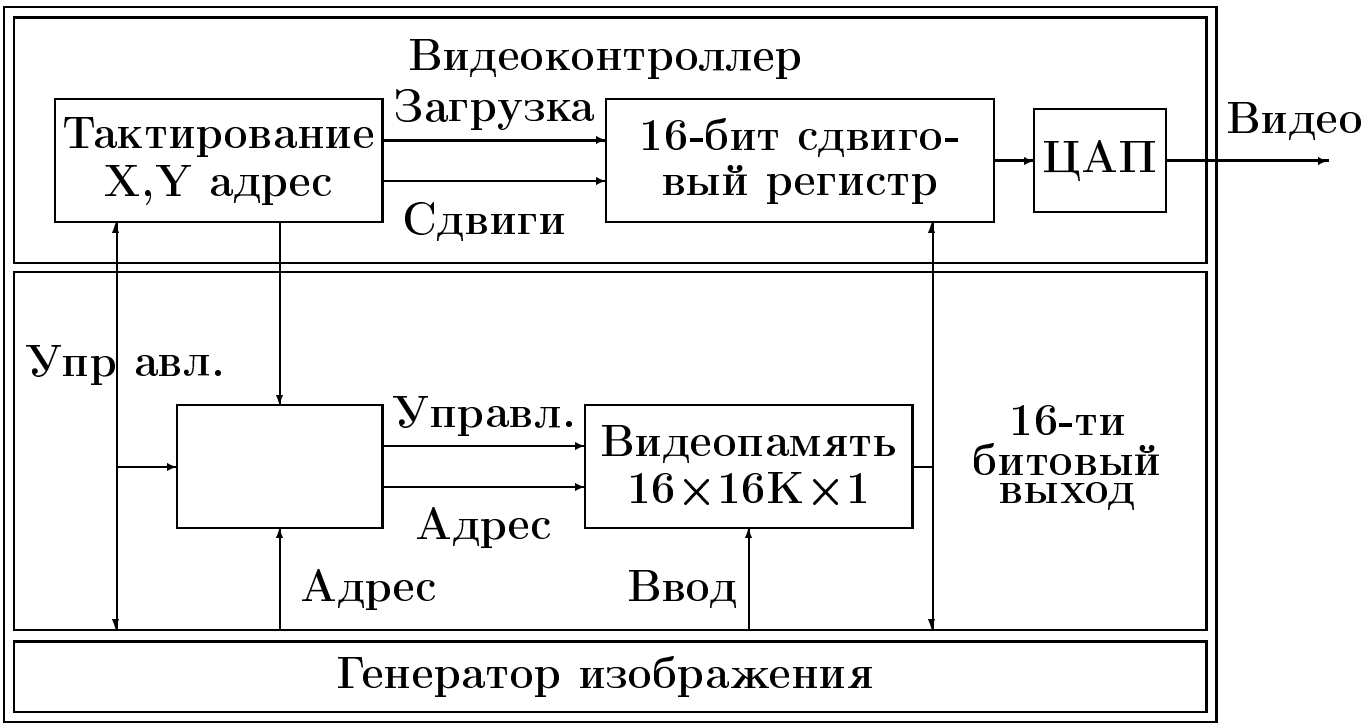
Характерные параметры дисплеев

Разрешение X×Y	f кадр. Гц	t полн. стр. мкс	t обр. стр. мкс	t обр. кадр. мкс	tп на пикс. нс	VОЗУ бит
512×512	30	60.4	10.9	1203	96.7	262 144
1024×768	60	20.92	4	600	16.52	786 432
1024×1024	60	15.69	4	600	11.42	1 048 576
1280×1024	60	15.69	4	600	9.13	1 310 720

Дисплей 1024×768, DRAM 64K×1, t = 340 нс.

340/16.52≈21 $\Longrightarrow$ VОЗУ=21×64×1024 = 1 376 256 бит.

На самом деле выбор будет: 24×64×1024 = 1 572 864 ровно в 2 раза больше чем надо!



Одновременный доступ к 16 чипам ∼ 1547 нс отображения.

Если DRAM 340 нс, то видеовывод занимает ≈ 22%, остальное — генератору.

Хотим сделать как лучше

Вместо 16×16K×1 возьмем 4×64K×1.
Одновременно запускаем 4 чипа.
Время отображения станет ∼ 387 нс.
Если DRAM 340 нс, то видеовывод займет уже ≈ 88%!
От применение микросхем большего объема стало хуже!

Решение проблемы недостаточного быстродействия:

- использование двойной буферизации;
- использование DRAM улучшенных технологий, например FPM DRAM;
- использование DRAM другой организации вместо nK×1 mK×4 или pK×8;
- использование памяти другой технологии, например, статической памяти;
- использование специализированных микросхем видеопамяти (VRAM, WRAM, SGRAM и т.д.).

Использование двойной буферизации

Дорогостоящее, технически сложное решение. Из одного кадрового буфера ведется отображение, второй отдается генератору изображений.

Работа со свободным от видеовывода буфером кадра:

- 1/2 периода кадровой развертки — актуализация изображения,
- 1/2 периода кадровой развертки — обновление изображения,
- смена буферов — в обратном ходе кадровой развертки.

Использование улучшенных DRAM

DRAM — для доступа к биту надо задать адрес строки, затем столбца;

FPM DRAM (Fast Page Mode DRAM) адрес строки задается один раз для нескольких доступов к близким элементам памяти. Такт 25–33 МГц, ширина полосы 80 Мб/с;

EDO DRAM (Extended Data Out DRAM) — наиболее употребимый тип DRAM. По сравнению с FPM DRAM модифицированы схемы тактирования за счет чего новое обращение к памяти может начаться до завершения предыдущего. Такт 40–50 МГц, ширина полосы 105 Мб/с.

BEDO DRAM (Burst Extended Data Out DRAM) — усовершенствование обычной асинхронной RAM. В BEDO DRAM EDO-память скомбинирована с конвейерной технологией и специальными триггерами с защелкой, что позволяет заметно сократить время доступа. BEDO DRAM по скорости приближается в SDRAM (см. ниже).

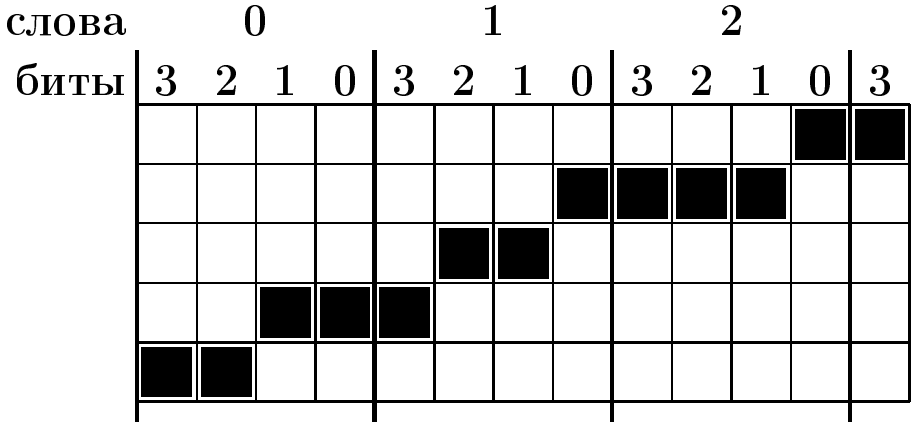
SDRAM (Synchronous DRAM). В обычной DRAM доступ к памяти ведется асинхронно относительно системного тактирования. Доступ в SDRAM синхронен с циклом тактирования. Это позволяет (после начальной задержки) в монопольном режиме 1 раз за такт иметь доступ к памяти. Такт 66–100 МГц, ширина полосы 166–253 Мб/с.

RDRAM (RAMBus DRAM). Ширина доступа 8 бит вместо обычных 32 или 64. 8-pp шина — не набор проводов, а скоростной интеллектуальный канал. Такт 250 МГц, ширина полосы 206 Мб/с.

Использование DRAM другой организации

Всесто микросхем nK×1 используются микросхемы nK×t, например, 16K×4, 32K×8. Варианты использования:

- в одном слове микросхемы t однобитовых пикселов. Сокращение времени доступа на видеовывод пропорционально t, но заметно замедляется занесение отдельной точки:



- В одном слове микросхемы t-битовый пиксел.
  Пусть дисплей 1024×1024×4.
  Видеопамять 64 микросхемы 64K×1.
  Пусть надо одновременно запускать 16 микросхем.
  Тогда за 1 доступ получим 4 пиксела по 4 бита.
  Заменим видеопамять 64×64K×1 на 64×16K×4.
  Тогда за 1 доступ получим 16 пикселов по 4 бита.
  Показатели улучшены.
  Для побитовой работы отдельные слои защищаются маской:

  V:= ((V OR M) XOR M) OR (N AND M)

  ((V OR M) XOR M): Зачистка разрядов в пикселе видеопамяти

  (N AND M): Вырезание по маске разрядов в заносимом пикселе

  V — старый пиксел; M — маска;
  N — новый пиксел.

Использование статической памяти

Статическая память (SRAM — Static RAM) заметно быстрее, так как использует триггеры в качестве элементов памяти. В PC SRAM используется для кэша уровня 1 или 2. В DRAM на один бит требуется один транзистор и конденсатор. В SRAM на один бит требуется четыре–шесть транзисторов. Поэтому SRAM дороже, потребляет больший ток и больше по размерам (при равной с DRAM емкости площадь кристалла примерно в четыре раза больше). При некоторых эксплуатационных требованиях (повышенная радиационная стойкость) использование SRAM — единственное решение.

Сравнение динамической и статической памяти

	Тип	Объем бит	Цикл нс	P, мвт доступ/ хранение	Корпус
К565РУ5	DRAM	64Kx1	240-450	360/30	16
К132РУ10А	SRAM	64Kx1	75	460/165	22

Использование специализированной видеопамяти

VRAM (Video RAM) — 2-портовая динамическая память, включающая:

- массив памяти nK×1 или mK×4;
- 256–2048-битовый сдвиговый регистр.

Сдвиговый регистр загружается строкой из массива памяти за один цикл (менее 1.5% полного времени) после этого он независим от массива памяти и может тактироваться с требуемой частотой.

Другими словами VRAM готова одновременно и для видеовывода и для генерации изображения.

VRAM TMS4161 фирмы Texas Instruments.

WRAM (Window RAM) — модификация VRAM, предложенная фирмой Samsung Electronics специально для использования в графических картах. Дополнительно поддерживает адресацию больших блоков (окон) видеопамяти и высокоскоростной обмен с памятью для обычно употребимых операций (заполнение блока, вывод текста). Ширина полосы пропускания выше на 25% и стоимость бита ниже на 20% чем у VRAM.

SGRAM (Synchronous Graphics RAM) — однопортовая синхронная память, имеющая значительно большую ширину полосы чем DRAM и специальные встроенные средства повышения скорости, разработанные для возможностей ускорения видеообработки в видеокартах.

MDRAM (Multibank DRAM) — однопортовая память, разбитая на отдельные блоки размером 32 Кб, к которым можно обращаться независимо. Такая организация памяти дает следующие возможности:

- intreleaving (чередование) доступа к банкам с целью их наложения по времени, что дает повышения скорости без использования двух портовой памяти;
- гибкое регулирование размера видеопамяти квантами по 32 Кб, а не по 1 или 2 Мб.
- Тактирование — 125–166 МГц, задержка — 22–19 нс, ширина полосы — 405–490 Мб/с.

Сравнение технологий видеопамяти

Приблизительное сравнение различных технологий приведено далее в таблице. Приблизительность состоит в том, что быстродействие памяти одной и той же технологии может значительно отличаться (до 2 раз), кроме того не всегда VRAM дороже EDO DRAM.

Технология	Портов доступа	Ширина полосы	Относ. цена	Ниша на рынке
DRAM	1	Малая	Малая	Малая
FPM DRAM	1	Малая	Малая	Малая
EDO DRAM	1	Малая	Малая	Малая
VRAM	2	Большая	Очень большая	Средне-большая
WRAM	2	Большая	Большая	Большая
SGRAM	1	Очень большая	Умеренная	Средняя
MDRAM	1	Очень большая	Большая	Средне-большая

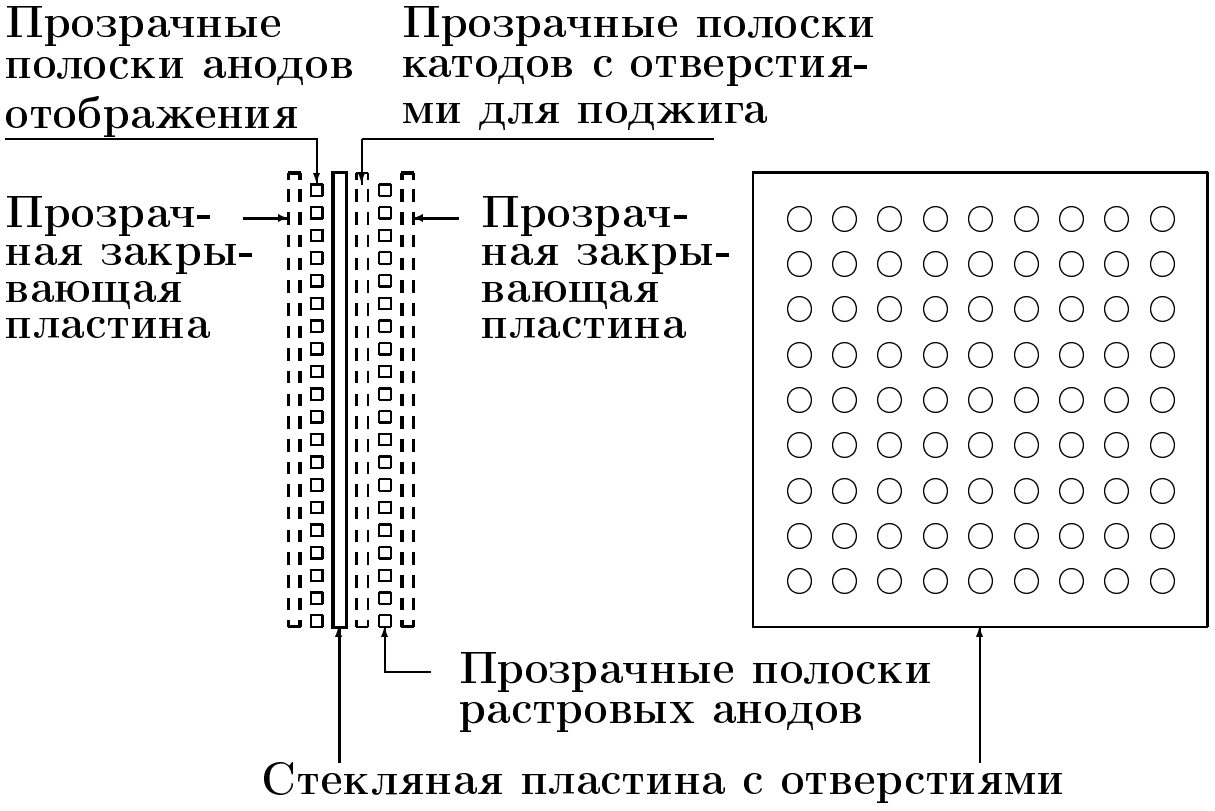
Дисплеи на альтернативных принципах

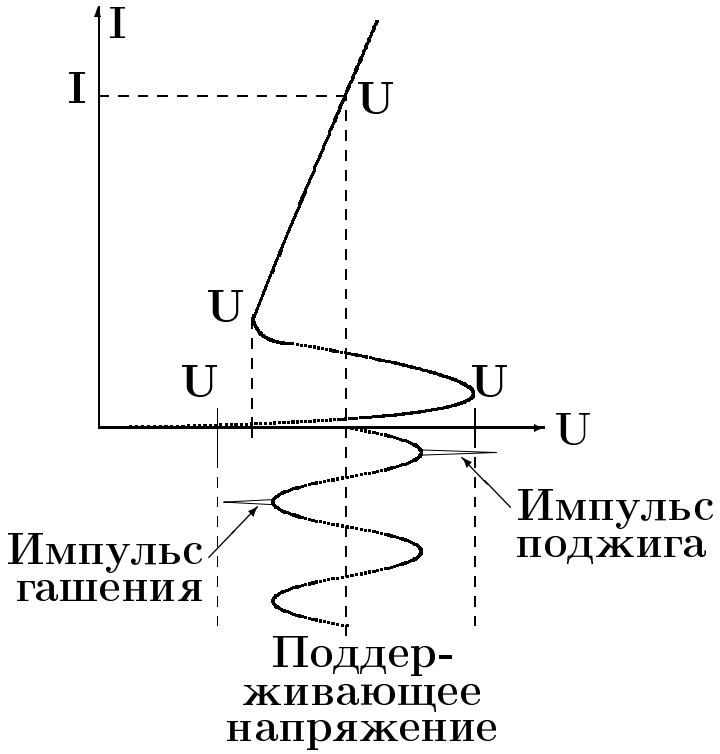
1. Плазменная панель.
2. ЖК-индикаторы.
3. Электролюминесцентные индикаторы.
4. Дисплеи с эмиссией полем.

Дисплеи с плазменной панелью

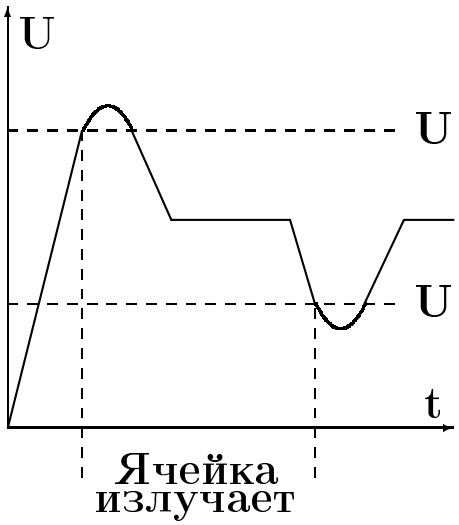
(Plasma Display Panels — PDP)

1966 — изобретена плазменная панель.
Используется свечение при тлеющем разряде в газе.
Разрешение $\approx$ 25 ячеек/сантиметр.
Время включения/выключения $\approx$ 20 мкс/точку.
Напряжение питания — десятки вольт.



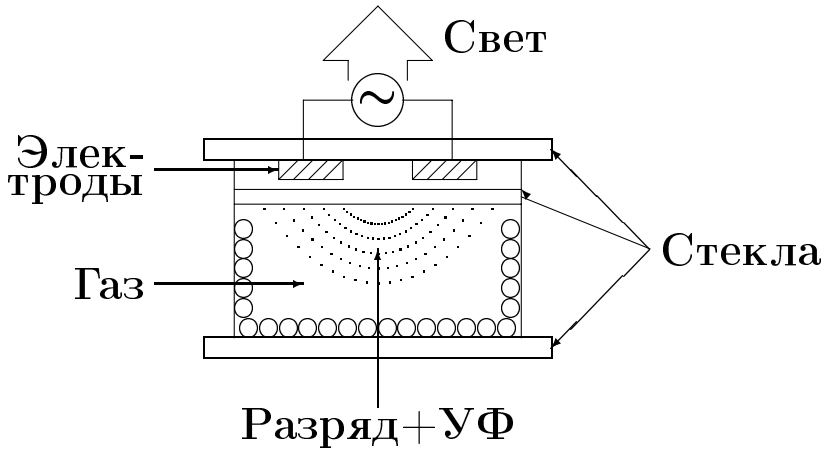


Вольт амперная характеристика разряда в газе



Временная диаграмма работы плазменной ячейки

Ячейка цветного дисплеи с плазменной панелью



Параметры дисплеев с плазменной панелью

- большой угол наблюдения (до 160°),
- изображение может запоминаться, выборочно стираться и строиться снова,
- поточечная адресация позволяет использовать как векторные, так и растровые принципы построения изображения,
- панель плоская, поэтому дисплей $\ll$ дисплея на ЭЛТ,
- изображение полностью лишено мерцания,
- информация от внешних источников изображений может проецироваться сквозь этот дисплей,
- соотношение цена/возможности хуже, чем у дисплеев на ЭЛТ,
- минимальное значение шага пикселов ~ 0.4 мм, (ЖК-дисплеи и дисплеи с эмиссией полем имеют шаг ~ 0.2 мм,
- относительно большое время включения/выключения ~ 20 мкс/точку,
- относительно высокое напряжения питания — десятки вольт,
- меньшая эффективность, так 40" дисплей потребляет ~ 300 Вт, такой же дисплей на ЭЛТ потребляет ~ 150 Вт, а пиковая яркость выше в 3 раза.

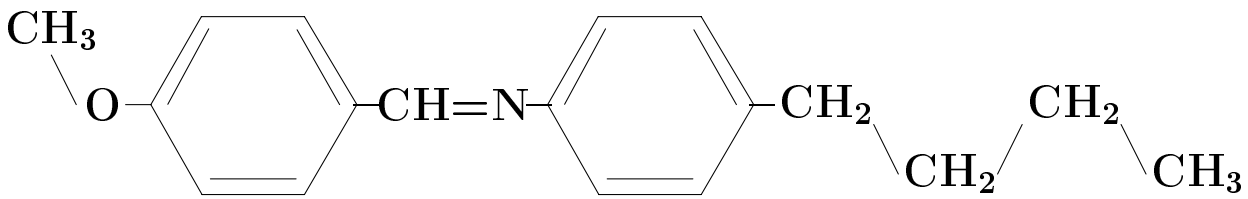
Параметры серийных дисплеев с плазменной панелью

Параметр	Хитачи		Фуджицу	
Диагональ	25"	41"	21"	42"
Разрешение	1024×768	1024×768	640×480	852×480
Шаг пикселов, мм		0.27 (гор.) 0.81 (вер.)		
Оттенков R×G×B		262 144 6×6×6	262 144 6×6×6	16.7млн 8×8×8
Контраст		300:1		
Яркость, кд/м^2			180	300
Угол обзора	160°	160°	160°	160°
Стандарты	XGA, SVGA, VGA, видео	XGA	VGA, видео	видео и данные
Экран, мм Толщина	508×381 80 мм	976×796 150 мм	422×314 32 мм	920×518 75 мм
Вес	15 кг	37 кг	10.6 lbs	40 lbs
Электропитание		100 Вт		

ЖК-индикаторы

1888 — ботаник Friedrich Reinitzer открыл ЖК.
1963 — Williams поляризационные эффекты в ЖК.
1973 — первый дисплей (EL 8025) на ЖК.

Структурная формула жидкого кристалла



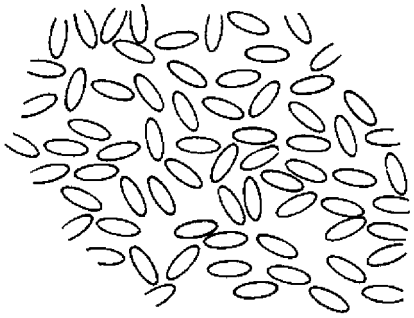
Фазы жидкого кристалла

изотропическая: позиция и ориентация молекул случайны;

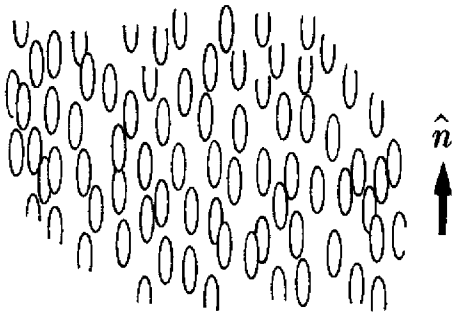
нематическая: позиции молекул случайны, ориентации одинаковы;

смектическая: молекулы дополнительно упорядочены в подвижных слоях;

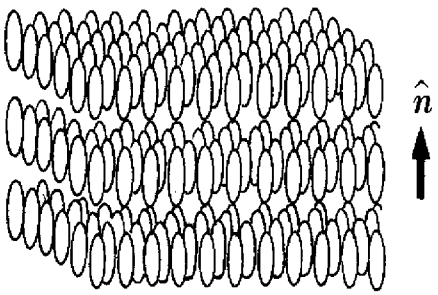
холеристиническая: молекулы упорядочены в тонких винтообразных слоях.



Изотропическая фаза

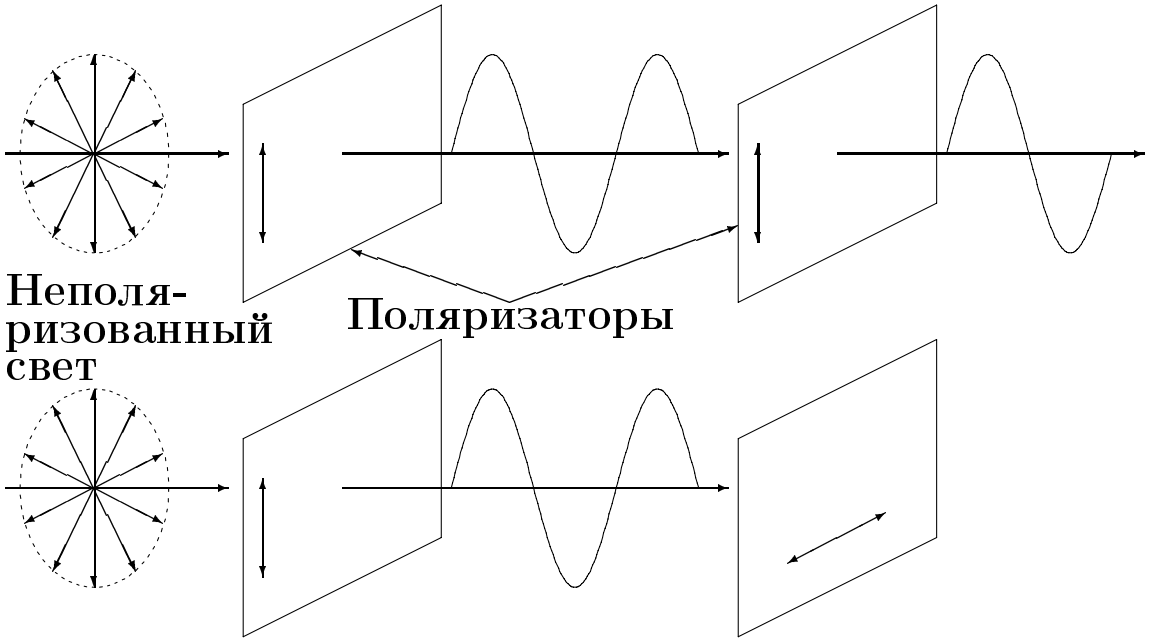


Нематическая фаза

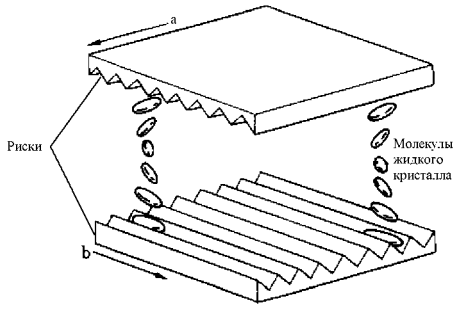


Смектическая фаза

Поляризация света



Закрученный нематический ЖК



Электро-оптическое переключение ЖК-кристаллом

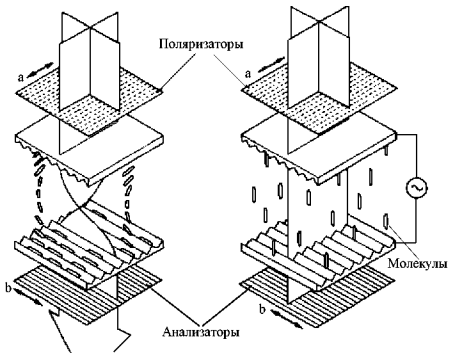
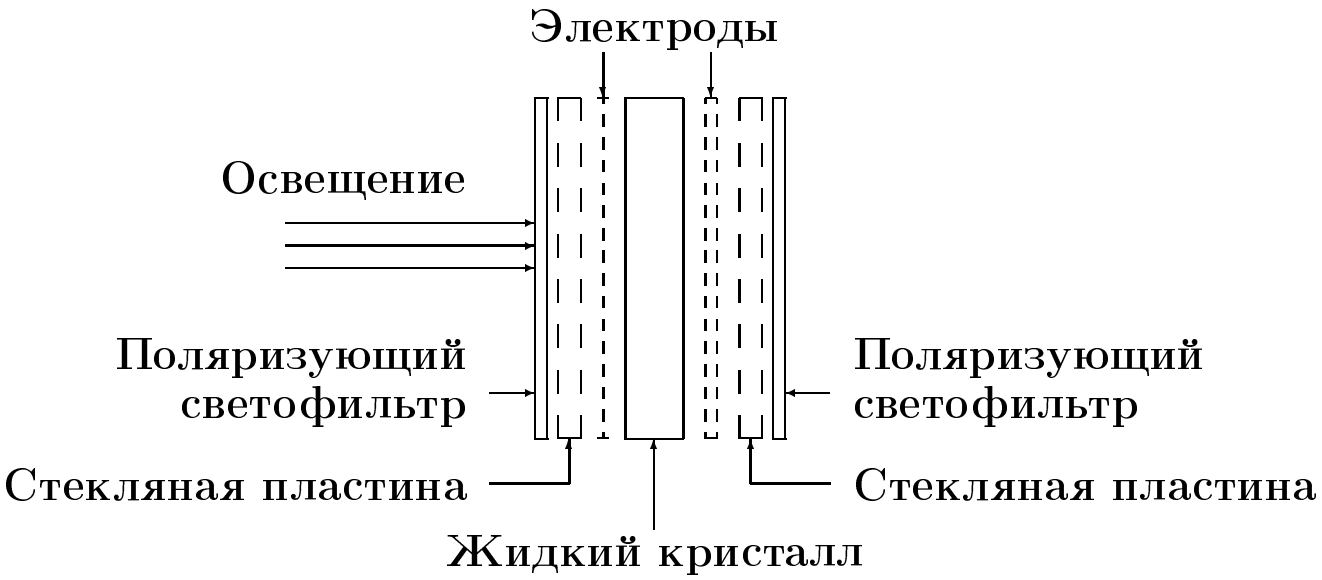


Схема ЖК-индикатора



Типы ЖК-индикаторов

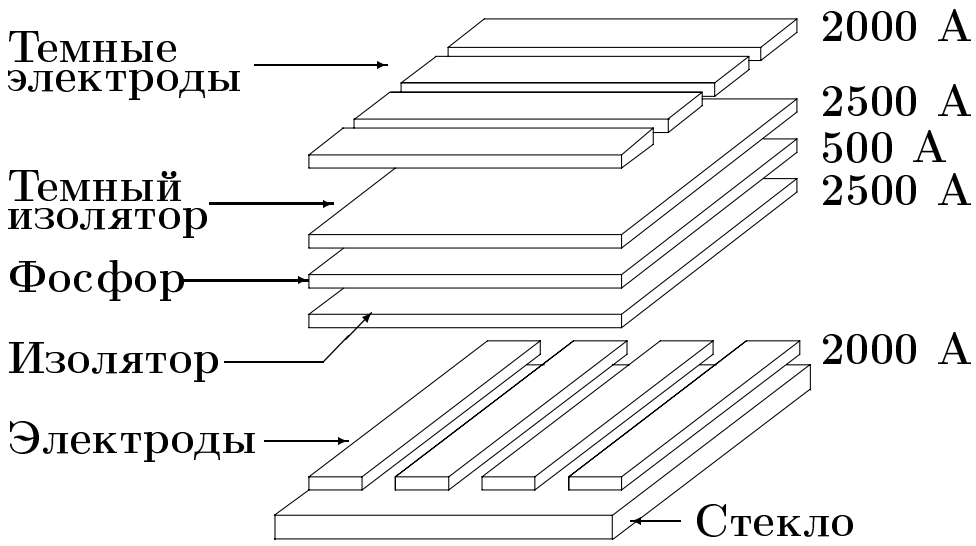
- с пассивной матрицей ($\sim$ 1 мс/строка);
- с активной матрицей — на каждый пиксел отдельный тонкопленочный транзистор (thin-film transistor, TFT).
  Цветной дисплей 800×600 — 1 440 000 TFT.
  Цена $\approx$ \$ 600.

Основные характеристики ЖК-индикаторов

- толщина $\sim 1/6$ ЭЛТ,
- вес $\sim 1/5$ ЭЛТ,
- энергопотребление $< 1/4$ ЭЛТ,
- отсутствует мерцание,
- отсутствуют геометрические искажения,
- отсутствует паразитное излучение,
- цена $\sim 3\times$,
- небольшая контрастность изображения $\sim$ 1:100,
- небольшая яркость $\sim 200\mathrm{cd/m^2}$,
- малый угол просмотра $\sim 50^\circ$,
- небольшая скорость работы,
- ограниченный температурный диапазон работы.

Электролюминесцентные индикаторы

1937 — открыта электролюминесценция (E $\sim 10^6$/);
1981 — начало практического использования.



Переменное напряжение, прикладываемое к электродам строк и столбцов, возбуждает свободные электроны в кристаллической структуре люминофора.

Эти электроны, сталкиваясь с атомами примеси, переводят их электроны на более высокие энергетические уровни.

При возврате в свое обычное состояние они испускают фотоны в видимом спектре.

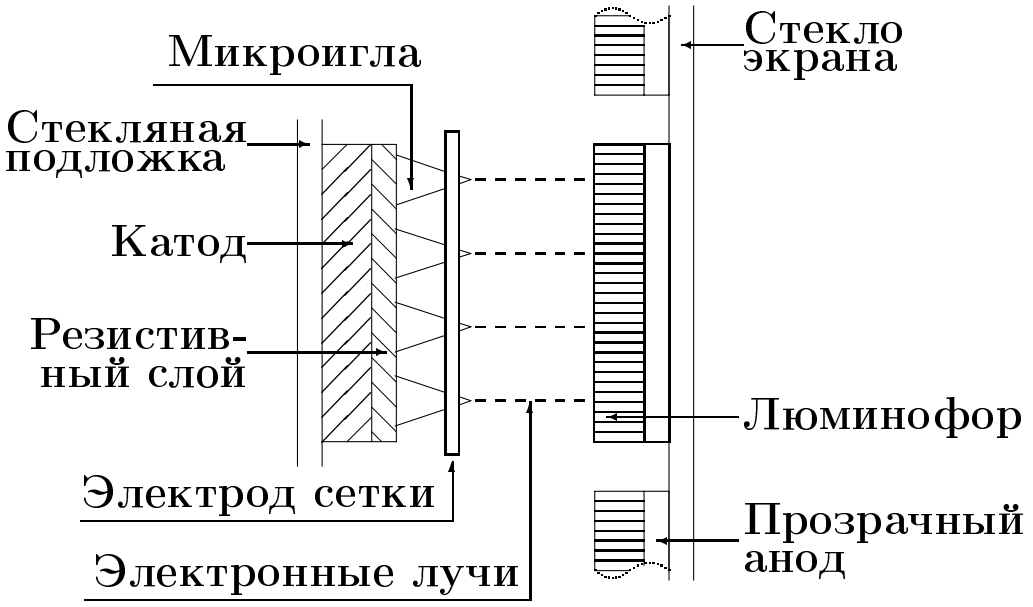
Основные параметры

- очень высокие контрастность и яркость,
- малая инерционность,
- небольшие размеры и высокая прочность,
- срок службы — более 120 000 часов,
- диапазон рабочих температур от -45°C до +65°C,
- угол обзора до 160°,
- люминофору требуется высокое напряжение (170–210 В).

Дисплеи с эмиссией полем

Дисплеи с эмиссией полем (Field Emission Display, FED) — плоские дисплеи, по принципу работы подобны обычным ЭЛТ.

Электроны генерируются из холодных катодов, имеющих форму очень острых микроигл, которых на каждый пиксел может иметься до нескольких тысяч.



Основные параметры

- небольшие габариты,
- широкий угол наблюдения ($\approx 180°$),
- малое энергопотребление (несколько ватт для дисплея размером с записную книжку),
- хорошее воспроизведение цветов (люминофор ЭЛТ),
- высокая скорость работы (та же что и ЭЛТ).

Дисплейная подсистема IBM PC

Основные компоненты — видеоадаптер и монитор.

Типы IBM PC видеоадаптеров:

MDA	Monochrome Display Adapter	640×350×1
HGC	видеоадаптер фирмы Hercules	720×350×1
CGA	Color Graphics Adapter	320×200×4
EGA	Enhanced Graphics Adapter	640×200×16
VGA	Video Graphics Array	800×600×256
8514/A		1024×768×256
XGA	eXtended Graphics Array	1024×768×256
SVGA	Super VGA — улучшенные версии VGA	
UVGA	Ultra VGA — улучшенные версии VGA	
TIGA	Texas Instruments Graphics Architecture	1280×1024×64 млн

Типы IBM PC мониторов:

композитный	на вход подается композитный NTSC-сигнал. Использовался с CGA.
цифровой	цифровой сигнал подается по N проводам. ЦАП находится в мониторе. Color Display — 4 провода, 16 цветов. Использовался с CGA, EGA. Enhanced Color Display — 16 цветов из 64. Многочастотный цифровой 16 цветов из 64 с различными частотами кадров. Использовался с EGA, VGA.
аналоговый	на монитор выдаются готовые RGB-сигналы. Используется с VGA.

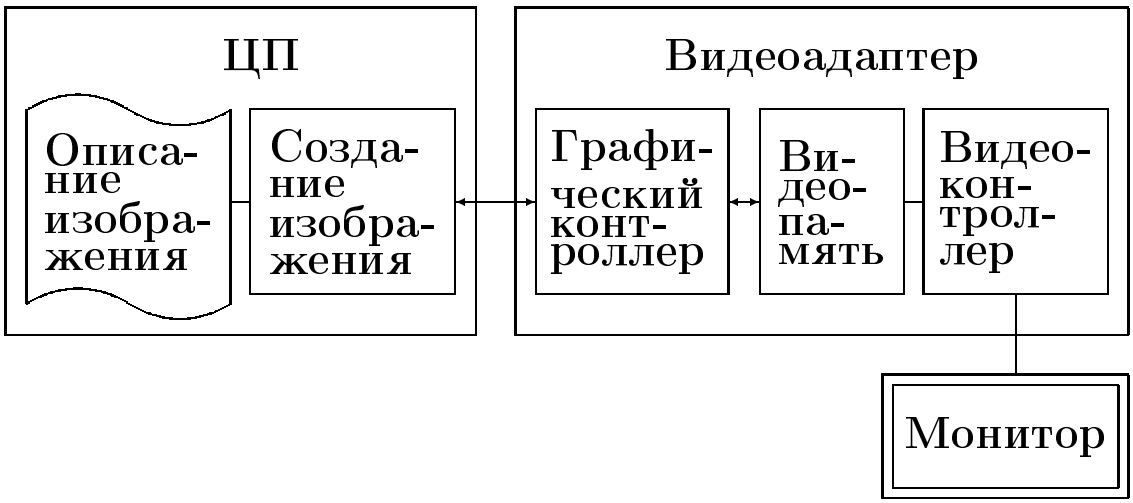
Характеристики видеоадаптеров IBM PC

Адаптер	Подключение	Разрешение	Развертки		Цветов
			Кадр., Гц	Строчн., кГц	
MDA	TTL	720×350	50	18.43	—
HGC	TTL	720×348	50	18.43	—
CGA	RGB/TTL	640×200	60	15.75	4
EGA	RGB/TTL	640×350	60	21.85	16
EGA+	RGB/TTL	640×480	60	30.50	16
VGA	RGB аналог	640×480	60/70	31.47	16
8514/A	—||—	1024×768	87 черес.	35.50	256
XGA	—||—	1024×768	72	37.50	256
SVGA-1	—||—	800×600	56	35.20	256
SVGA-2	—||—	800×600	60	37.80	256
SVGA-3	—||—	800×600	72	48.30	256
UVGA-1	—||—	1024×768	60	48.40	256
UVGA-2	—||—	1024×768	70	56.50	256
UVGA-3	—||—	1280×960	60	64.00	256
TIGA	—||—	1280×1024	60	64.00	64 млн

Глубина цвета — количество бит на пиксел:

Наименование	Глубина	R×G×B	Цветов
High Color	15	5×5×5	32 768
Direct Color	18	6×6×6	262 144
	21	7×7×7	2 097 152
True Color	24	8×8×8	16 777 216

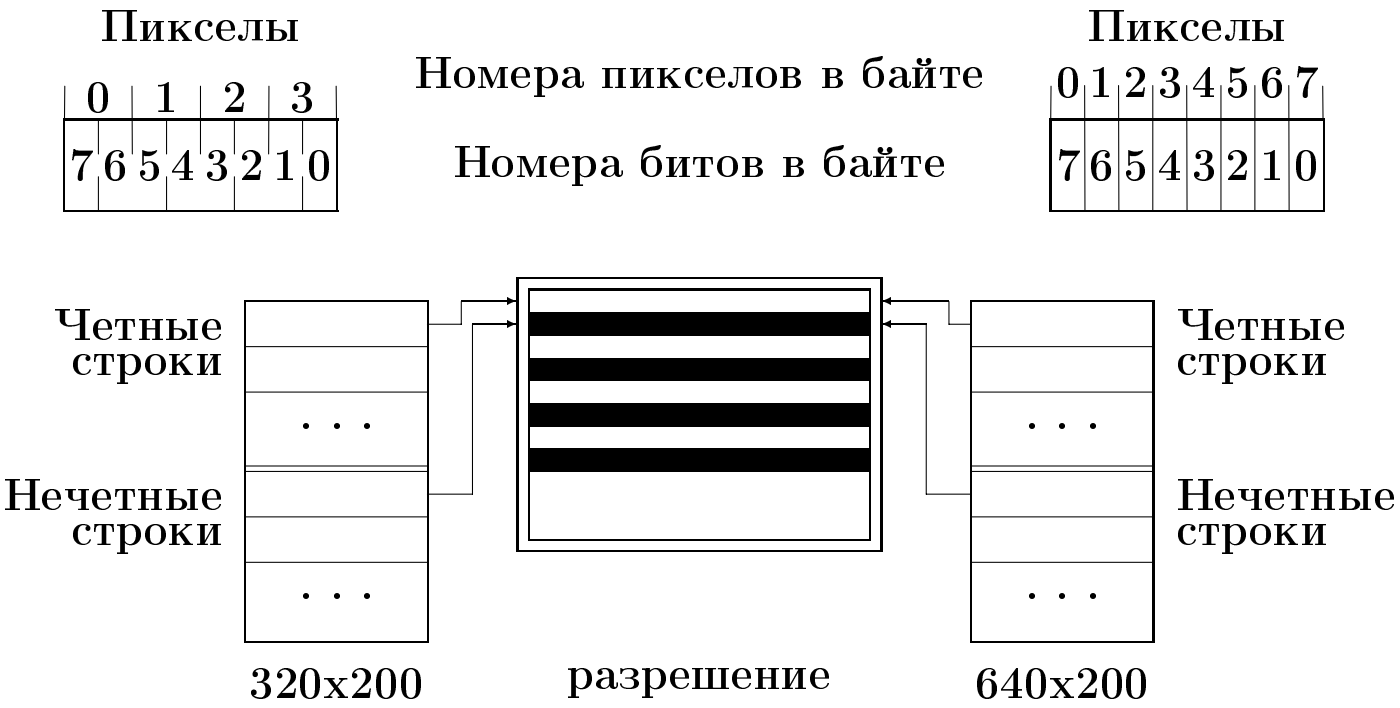
Архитектура дисплейной подсистемы IBM PC



Видеопамять доступна ЦП как обычная оперативная память и размещается в адресном пространстве с A000:0000 по B000:FFFF.

Два режима — символьный и графический.

Видеоадаптер CGA

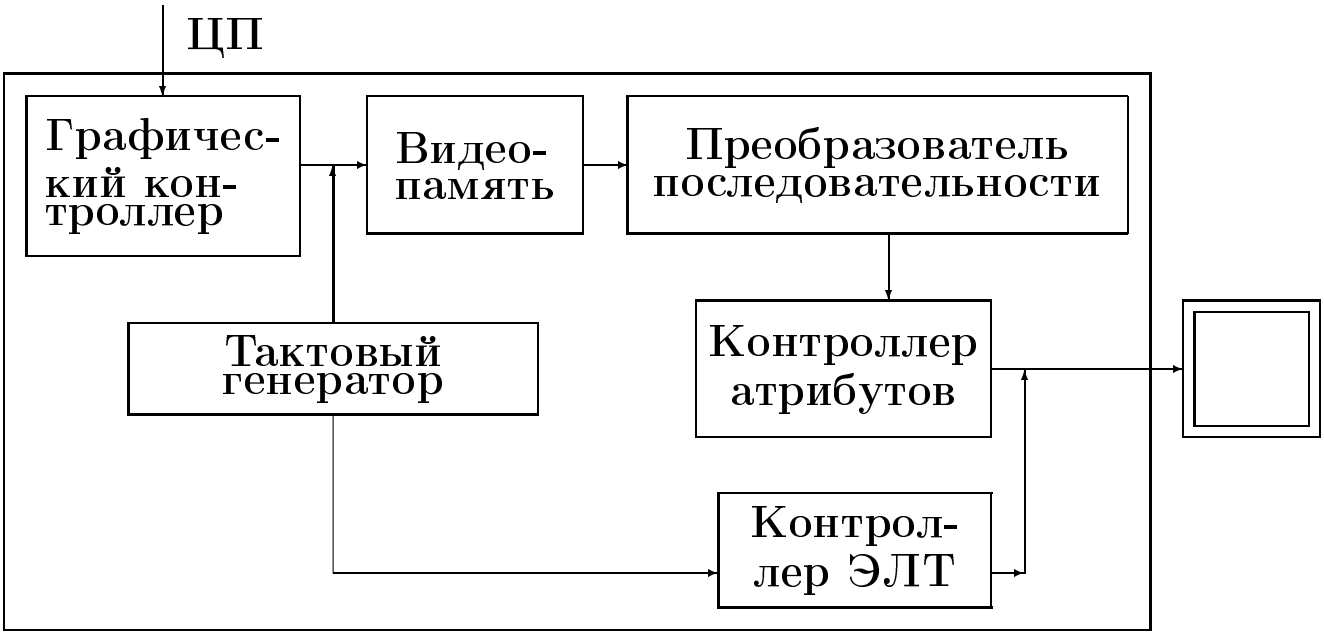


Видеопамять 16 Кбайт.

Четные строки — B800:0000, нечетные — B800:2000.

Символьный режим 8×8, 2 байта на символ.

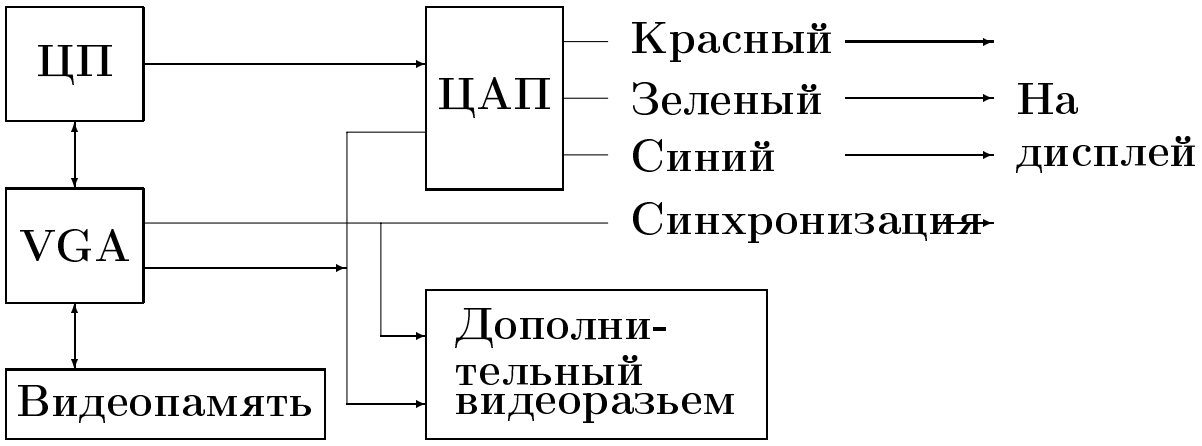
Видеоадаптеры EGA, VGA



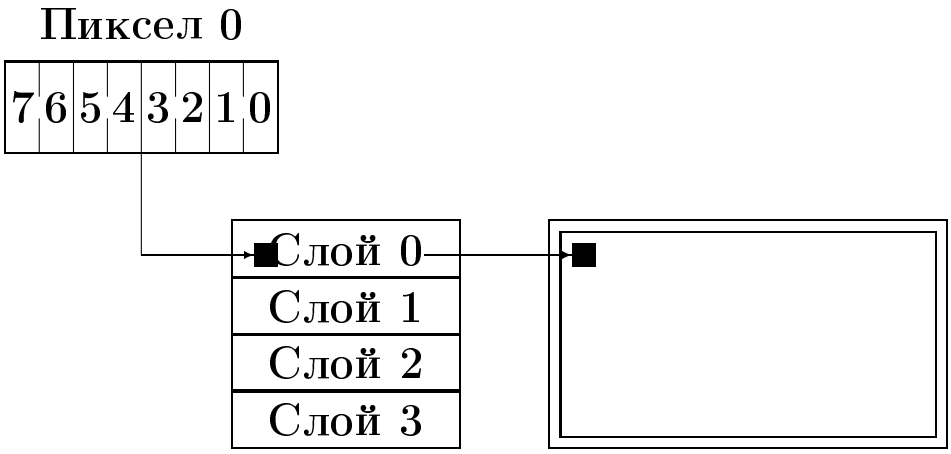
Видеопамять разбита на 4 банка (слоя).
По каждому адресу расположено 4 байта.
Видеопамять может состоять из нескольких страниц.

Текстовый режим — 25, 43 (EGA) и 50 (VGA) строк.
На символ 2 байта.
1-й — код символа, 2-й — атрибут.
Таблицы знакогенератора размещены во 2-м слое.
Может иметься 2 таблицы.

Видеоадаптер VGA



Структура видеопамяти для VGA-режима 256 цетов



Графические режимы.

CGA — используется только 0-й слой памяти.
Послойная память (1, 2, 4 слоя).
Линейная последовательность всех слоев (режим 256 цветов для VGA).

Устройства ввода

- Клавиатуры,
- Кнопки,
- Световое перо,
- Планшеты,
- Мышь, трекбол, джойстик,
- Потенциометр,
- Растровый сканер,
- Кодировщик.

Клавиатуры (Keyboards)

Основные характеристики клавиатуры:

- метод обнаружения нажатия клавиши:
  - механическое замыкание контактов,
  - изменение емкости,
  - изменение магнитного поля,
  - прерывание луча света и т.д.;
- кодировка — ASCII (American Standard Code for Infomation Interchange), КОИ-7, КОИ-8 (ГОСТ), MIC (Болгария) и т.д.;
- количество клавиш редактирования текста;
- возможности расширения кодировки за счет нажатия дополнительных клавиш — верхнего и нижнего регистров (Shift), клавиш для задания управляющих символов и кодов (Ctrl, Alt);
- эргономические свойства — размеры и расположение клавиш, наличие тактильной обратной связи при нажатии и ощущение контакта при полностью нажатой клавише.

Кнопки (Buttons)

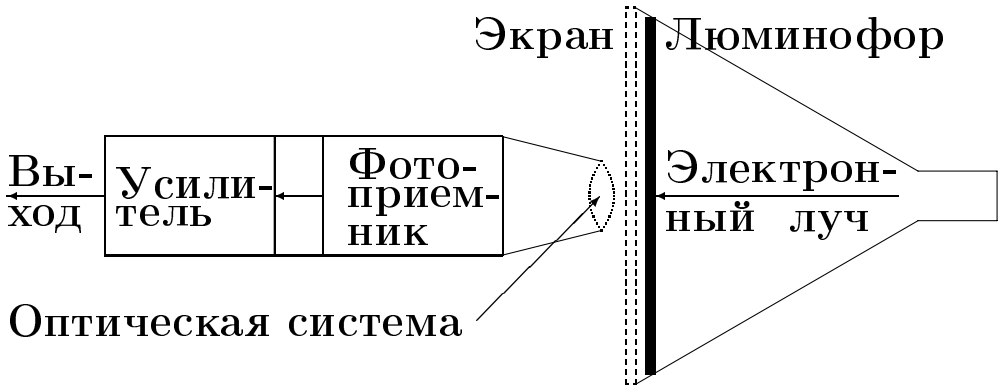
Служат для задания некоторого воздействия, связанного с нажатием, отпусканием или удержанием кнопки (ввод команды, выбор параметра).

Конструктивно кнопки устроены аналогично клавишам текстовой клавиатуры.

Основное отличие в использовании — с кнопкой не связано определенного воздействия, такого как появление символа.

Результат воздействия на кнопку программируется.

Световое перо (Lightpen)



Служит для непосредственного указания элементов изображения на экране.

В основном использовалось в векторных дисплеях, в которых имеется однозначное соответствие между отрабатываемой графической командой дисплейного файла и элементом изображения.

Указанный элемент обычно выделяется сверхподсветкой.

В растровых дисплеях световое перо выдает координату указанной группы пикселов.

Для установления соответствия между указанным пикселом и элементом изображения требуется программная интерпретация дисплейного файла.

Для идентификации составных объектов используется техника идентификаторов указания, как правило совместно с программной интерпретацией дисплейного файла.

Пример дисплейного файла с двумя указуемыми объектами

Команда	Смысл	Стек picid
. . . . .	любые дислейные команды	пусто
picid 001	идентификатор указания 001	001
. . . . .	графические команды	001
. . . . .	построения элемента 001	001
end_picid	конец указуемой группы	пусто
. . . . .	любые дислейные команды	пусто
picid 002	идентификатор указания 002	002
. . . . .	графические команды	002
. . . . .	построения элемента 002	002
end_picid	конец указуемой группы	пусто
. . . . .	любые дислейные команды	пусто
endpic	конец дисплейного файла	пусто

Планшеты (Tablets)

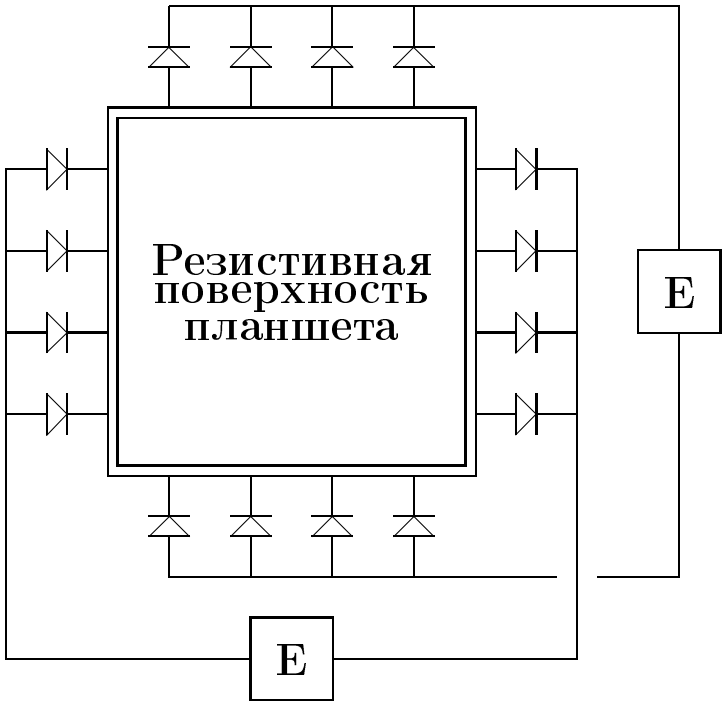
Устройства непосредственного ввода координат (локаторы).

Позиции задаются перемещением зонда планшета (визира или карандаша) по рабочей поверхности. Координата текущего положения зонда определяется с частотой от 200 до 500 раз в секунду.

Режимы работы планшетов:

- точечный — координата генерируется при нажатии кнопки зонда;
- непрерывный — последовательность координат генерируется непрерывно при нахождении зонда в рабочей области планшета;
- переключаемый непрерывный — непрерывная последовательность координат генерируется при нажатии кнопки зонда;
- приращений — генерируются приращения к последней выданной позиции.

Потенциометрический (градиентный) планшет

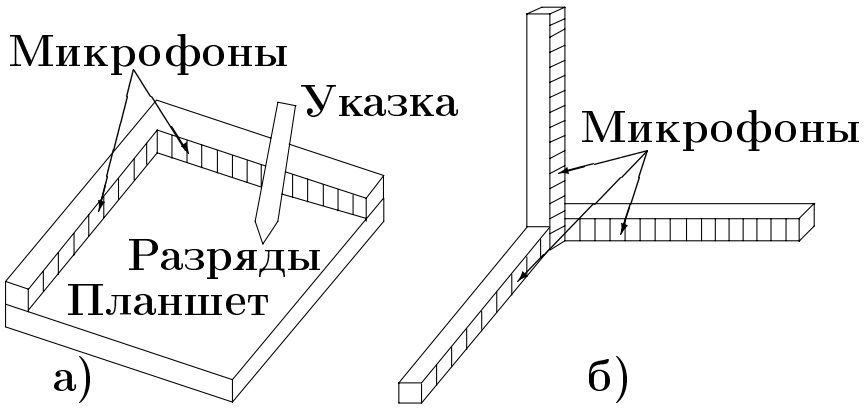


Зонд планшета имеет гальванический контакт с его резистивным покрытием.

Акустический планшет

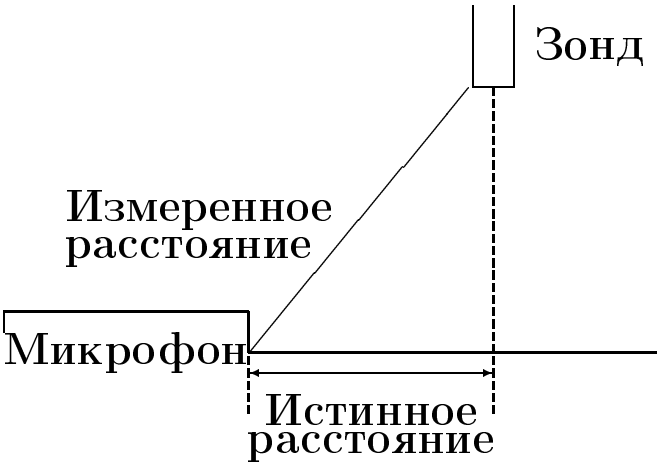
Разрядник в зонде излучает ультразвуковой сигнал, который принимается ленточными микрофонами, расположенными на двух смежных сторонах планшета (рис. а).

Акустические планшеты с тремя группами микрофонов могут выдавать трехмерную координатную информацию (рис. б)

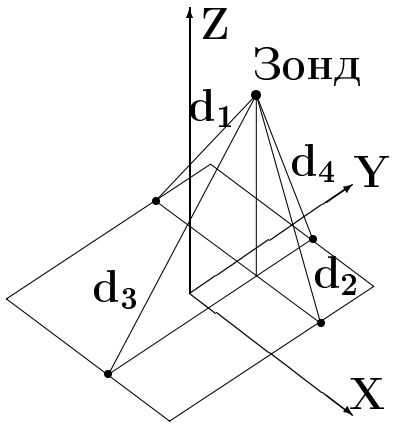


Особенности акустического планшета

Как показано на рис. а) при большом удалении зонда от поверхности координаты определяются с ошибками



а)



б)

Использование микрофонов по каждой из 4 сторон планшета позволяет измерять z-координату подъема зонда над поверхностью планшета. При стороне планшета = 2a и измерении координат x и y относительно центра планшета из рис. б) можно написать следующие выражения для z:

$$\begin{aligned} z^2 &= d_1^2 - (x+a)^2, \\ &= d_2^2 - (x-a)^2, \\ &= d_3^2 - (y+a)^2, \\ &= d_4^2 - (y-a)^2. \end{aligned}$$

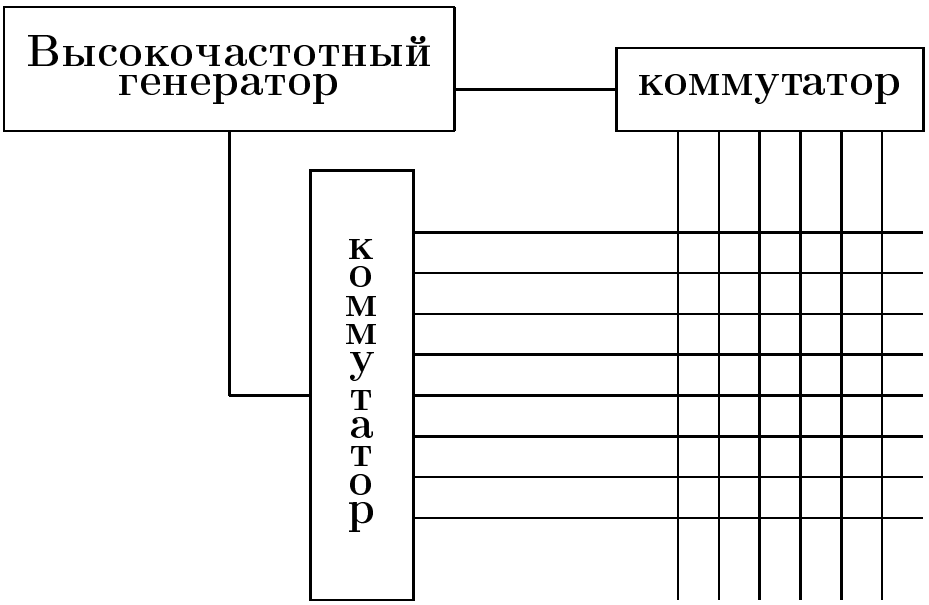
Из этого легко находятся:

$$x = (d_1^2 - d_2^2) \;/\; 4a;$$

$$y = (d_3^2 - d_4^2) \;/\; 4a;$$

$$4z^2 = (d_1^2 + d_2^2 + d_3^2 + d_4^2) + 2x^2 + 2y^2 + 4a^2.$$

Емкостные планшеты



Антенны

Под непроводящей рабочей поверхностью генерируется электромагнитное поле с помощью взаимно перпендикулярных групп излучающих проводников-антенн.

Проводники в каждой группе должны быть точно параллельны и находиться на одинаковых расстояниях друг от друга.

На эти проводники поочередно подается высокочастотное напряжение. Сигнал принимается емкостным датчиком зонда, который не может находиться от поверхности планшета на расстоянии большем чем толщина нескольких листов бумаги. По соотношению амплитуд сигналов, принятых от двух ближайших проводников, вычисляется расположение зонда.

Пример планшета: рабочее поле — 380×380 мм, проводники расположены на двухсторонней печатной плате с шагом 5мм, частота генератора — 625 КГц, время коммутации — 200 мкс, датчик — незамкнутая круглая проволочная петля, точность определения позиции — 0.1 мм, время вычисления позиции — 7 мс.

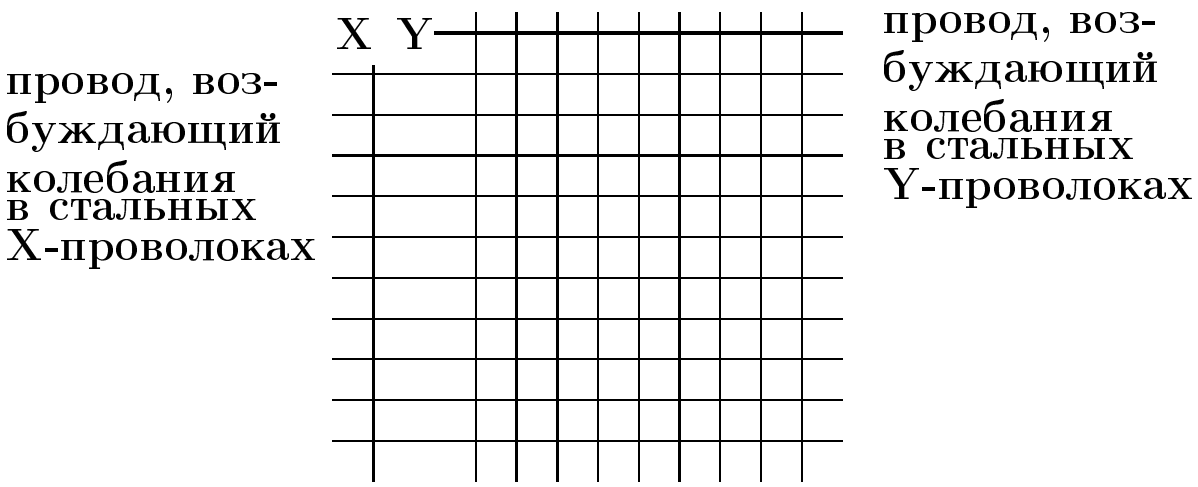
Магнитоэлектрические планшеты

Устроены подобно емкостным, но датчик зонда — катушка. Проводники под рабочей поверхностью и катушка зонда служат обмотками трансформатора.

При использовании катушки зонда как приемника параметры такого планшета близки к параметрам емкостного.

Существенно большее разрешение достигается при использовании многовитковой обмотки зонда как передатчика. Такой вариант используется в кодировщиках для ввода чертежей.

Магнитострикционные планшеты



Магнитострикционный эффект — незначительное изменение формы материала под воздействием внешнего магнитного поля.

Магнитное поле, генерируемое передающими катушками на краю планшета и перпендикулярное магнитострикционным проволокам, вызывает изменение их длин.

Это изменение длины распространяется вдоль проволоки со скоростью около 5000 м/с.

Волна, попадая в приемную катушку, расположенную в зонде планшета, формирует в катушке импульс напряжения.

Время прихода волны пропорционально расстоянию от передающей катушки до зонда. Т.к. расстояние всегда измеряется вдоль проволоки, то не требуется, чтобы проволоки были абсолютно параллельны.

Планшеты на этом принципе имеет относительно высокую точность (0.01 мм), широко используется в робототехнике и компьютерной графике.

Пример планшета (Bit Pad One фирмы Summagraphics): рабочее поле — 300×300 мм, шаг проволок ≈ 3 мм (по 96 шт по каждой из осей). разрешение по координате — 0.1 мм.

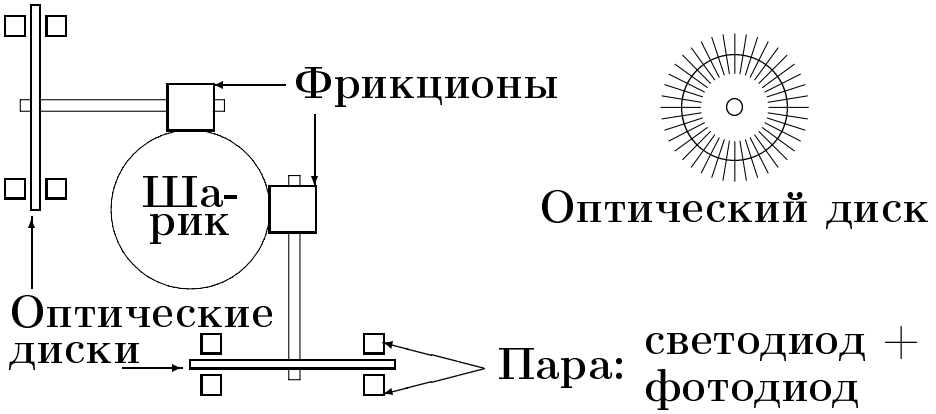
“Мышь” (Mouse), трекбол (Treckball), джойстик (Joystick)

Не прямой ввод графической информации, как правило, основан на перемещении пользователем по экрану графического курсора.

Методы позиционирования курсора:

- статический абсолютный,
- статический относительный,
- динамический (направление и скорость перемещения курсора задаются установкой устройства ввода, например, направление перемещения задается направлением отклонения рычага, а скорость перемещения — величиной угла отклонения.

“Мышь”



Перемещение “мышки” произвольное и не слишком точное, поэтому наиболее подходящий метод использования — статический относительный,

Трекбол

Трекбол — перевернутая “мышка” с одним большим шаром, приводимым в действие рукой. При этом имеется хорошая тактильная обратная связь, поэтому для управления курсором может использоваться не только координата, но и угол поворота шара.

Так как шар достаточно тяжелый, то можно использовать и величину начального импульса шара.

Т.е. трекбол применим для всех трех способов позиционирования курсора — статического абсолютного, статического относительного и динамического.

Джойстик

Джойстик — вертикально стоящий рычаг, который на нижнем конце установлен в кардане, удерживается пальцами в среднем (начальном) состоянии и может отклоняться одновременно по двум осям. Перемещения передаются на два потенциометра, выдающих напряжения для каждой их координат.

Очень простая и дешевая конструкция джойстика — с жестко закрепленным жезлом, к которому прикреплены датчики растяжения, например, пьезоэлектрические.

Основное применение джойстика — динамическое позиционирование, однако он применим и для статического абсолютного очень быстрого, но неточного позиционирования.

Потенциометр

Потенциометр с аналого-цифровым преобразователем обеспечивает быстрый ввод точных числовых значений.

Используются потенциометры вращения и ползунковые.

Растровый сканер

Используются для растрового ввода изображений.

Одна из важных областей применения — ввод текстов. При этом обработка введенного изображения выполняется с программного обеспечения распознавания текстов (Optical Character Recognition — OCR).

В САПР сканеры используются для автоматизации ввода ранее подготовленной конструкторской документации. В этом случае проблема состоит в необходимости выполнения обратного преобразования растр-вектор. Необходимо распознавать различные изображения и тексты в том числе рукописные, учитывать, что изображение представляется поточечно, причем одна и та же линия может получить при сканировании не только различную ширину, но и дырки и т.д.).

Сканеры работают в один или три прохода, большинство сканеров работает в один проход.

Простейшие сканеры — ручные с шириной сканирования до 127 мм (5 дюймов) и разрешеним 100, 200, 300 или 400 точек на дюйм.

Стационарные растровые сканеры

Основные варианты стационарных сканеров:

1. Оригинал перемещается относительно неподвижной линейки фотоприемников (сканеры с полистовой подачей и барабанные сканеры для больших форматов).

2. Линейка фотоприемников перемещается относительно оригинала (планшетные сканеры).
3. Проекционные сканеры, в которых изображение неподвижного оригинала проецируется на матрицу фотоприемников, установленных в фокусе объектива. Объектив же перемещается для выбора нужного фрагмента с нужным увеличением.

Сканирование цветных изображений обеспечивается сменой светофильтров. В некоторых сканерах смена производится вручную.

Разрешение стационарных сканеров от 300×300 точек на дюйм (dot per inch — dpi) до 1200×1200 dpi. С использованием интерполяции разрешение достигает от 4800×4800 dpi до 10000×10000 dpi.

Поддержка цветов в стационарных сканерах — либо серая шкала, либо 24 бита/пиксел ($>$ 16 миллионов), либо 30 бит/пиксел ($>$ 1 миллиарда) и до 36 бит/пиксел ($>$ 68 миллиардов).

Выходные форматы растровых файлов — TIF, GIF, BMP, PCX и т.д. Интерфейс для подключения к ПЭВМ — либо параллельный порт, либо SCSI. Обычно сканеры имеют встроеннный буфер для сохранения изображения объемом от 32 K до 2 M. Сканеры с полистовой подачей обеспечивают ввод и сканирование от 3 до 8 листов в минуту (при одном проходе).

Кодировщик

Кодировщик (графоповторитель) — устройство полуавтоматического и/или автоматического ввода.

Полуавтоматические кодировщики, подобны большому (магнитоэлектрическому) планшету, но с высокоточным позиционированием зонда. Обычно имеется режим привязки позиций к задаваемой сетке.

Определение элементов документа выполняется оператором, который идентифицирует их с помощью обширной ФК.

Основные режимы работы:

- дискретный, когда оператор устанавливает зонд в требуемую точку и выдает команду определения координаты;
- непрерывный, когда оператор перемещает зонд вдоль некоторой линии, а устройство автоматически генерирует последовательность координат.

Автоматические кодировщики — комбинация высокоточного планшетного графопостроителя, оснащенного не пишущим узлом а считывающим зондом, и (магнитоэлектрического) планшета. Выполняет сканирование вводимого документа (чертежа, графика, карты), определение и считывание его элементов.

Известны реализации, использующие растровый сканер с последующей интерактивной программной обработкой документа.

Основные проблемы в этом случае — преобразование растр — вектор и недостаточные форматы вводимых документов, что требует решения задачи “сшивки” отдельно введенных фрагментов.

Нетрадиционные устройства

- различные (аккордные) клавиатуры с одновременным нажатием нескольких клавиш;
- сенсорная панель (Touch Screen);
- речевой диалог (ввод/вывод);
- space ball — комбинация мышки и трекбла;
- перчатки и костюмы данных (power glove, date glove, date suit);
- head mounted display с трассировкой наклона и положения головы и даже глаз.

Перчатка данных