Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования

**"САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика С.П. КОРОЛЕВА"**

Лабораторный практикум по курсу "Компьютерная графика"

(лабораторные работы № 1 и 2)

Методические указания

Самара 2005

Составители: *К.Е. Климентьев, М.А. Кудрина*

УДК 681.3

Лабораторный практикум по курсу "Компьютерная графика" (лабораторные работы № 1 и 2): Метод. указания / СНЦ РАН; Сост. К.Е. Климентьев, М.А. Кудрина. Самара, 2005. 32 с.

В методических указаниях рассмотрены необходимые теоретические сведения для выполнения лабораторных работ № 1 и 2 по курсу "Компьютерная графика". Также методические указания содержат индивидуальные задания.

Методические указания предназначены для студентов, обучающихся по специальности 230102 "Автоматизированные системы обработки информации и управления". Разработаны на кафедре информационных систем и технологий СГАУ.

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета имени академика С.П. Королева.

Рецензент: доцент кафедры информационных систем и технологий СГАУ, к.т.н. О.П. Солдатова.

Введение

Курс "Компьютерная графика" занимает важное место в системе подготовки студентов по специальности 230102 "Автоматизированные системы обработки информации и управления". Целью изучения дисциплины является получение студентами комплекса знаний и умений в области создания и использования графических интерфейсов АСОИУ. Задача курса состоит в изучении и практическом освоении студентами способов формирования, отображения, преобразования и хранения графической информации.

Лабораторный практикум направлен на получение студентами практических навыков в тематических рамках, охватываемых курсом "Компьютерная графика". Данные методические указания предназначены для выполнения лабораторных работ по темам "Использование технических возможностей графических контроллеров VGA/SVGA" и "Алгоритмы построения и преобразования изображений".

Лабораторные работы выполняются в режимах аудиторных и самостоятельных занятий. Работы должны быть выполнены в установленные преподавателем сроки. Форма отчетности по лабораторным работам: демонстрация отлаженной программы, обсуждение методов и средств ее реализации и результатов ее работы.

Порядок выполнения лабораторных работ

1. Ознакомиться с теоретической частью.
2. Получить задание у преподавателя.
3. Написать и отладить программу.
4. Продемонстрировать работу и текст программы преподавателю.
5. Ответить на дополнительные вопросы.

Примечание: в лабораторных работах **запрещается** использовать стандартные функции вывода на экран. Функции вывода символов (в текстовых режимах) и пикселов (в графических режимах) должны быть написаны **самостоятельно**.

Лабораторные работы выполняются в операционной системе MS-DOS (возможно использование DOS-сессии операционных систем Windows 95/98/ME). Языки программирования: Pascal или C. Рекомендуемые системы программирования: Turbo Pascal v6.0/7.0 и Borland C/C++ v3.1. Применение расширенных возможностей языка C++, а также использование языка ассемблера **не допускается**.

Если не указано иное, то текстовым видеорежимом по умолчанию следует считать 80x25x16, а графическим – 320x200x256. Каждое изображение, построенное в графическом режиме, должно обязательно сопровождаться комментарием на русском языке, выведенным поточечно.

Лабораторная работа № 1

Тема: ***"Использование технических возможностей графических контроллеров VGA/SVGA"***

Теоретическая часть

Видеоподсистема ПЭВМ состоит из двух устройств:

- *видеоконтроллер* (синонимы - *видеоадаптер*, *видеокарта*, *видеоплата*);
- *дисплей* (с электронно-лучевой трубкой (ЭЛТ) или жидкокристаллический).

Изображение на ЭЛТ-дисплее формируется в результате построчного перемещения по люминофору световой точки (луча). Типичное количество строк в телевизоре - 625, на компьютерных дисплеях от 200 до 2048. Электронный луч пробегает по порядку все строки пикселей. Поскольку после прекращения воздействия электронного луча на точку экрана ее свечение быстро затухает, то сканирование периодически повторяется - в зависимости от качества дисплея - от 60 до 120 раз в секунду. При прорисовке кадра на прямой ход луча затрачивается 80-95% общего времени.

Рис. 1. Перемещение луча по экрану

1. Устройство видеоконтроллера

Практически все современные видеоконтроллеры принадлежат к комбинированным устройствам и помимо своей главной функции - формирование сигналов, в соответствии с которыми монитор может отображать ту или иную информацию на экран, - осуществляют ускорение выполнения графических операций.

Видеоконтроллер состоит из следующих функциональных блоков: видеопамять, схема декодирования, цифро-аналоговый преобразователь (ЦАП) (см. рис. 2).

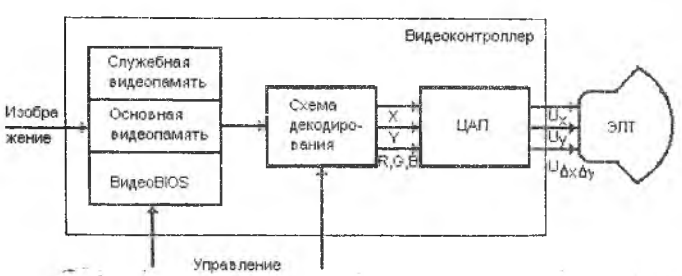


Рис. 2. Функциональная схема видеоподсистемы ПЭВМ

Видеопамять- это набор ячеек памяти, делящийся на три части:

1) *основная* видеопамять;
2) *служебная* видеопамять;
3) видеоBIOS.

В технических характеристиках видеокарты обычно указывается объем основной видеопамяти, например: 16 Мб.

В *основной* памяти формируется «образ» изображения. В общем случае (это зависит от видеорежима) основная память разделена на несколько *страниц* и несколько *слоев* (см. рис. 3).

Одна из страниц может быть назначена *активной* (текущей), любое изменение информации на ней сразу же отображается на экране. Нулевой слой видеопамяти является частью адресного пространства ПЭВМ в районе адресов A000h:0 – B000h:FFFFh (см. табл. 1).

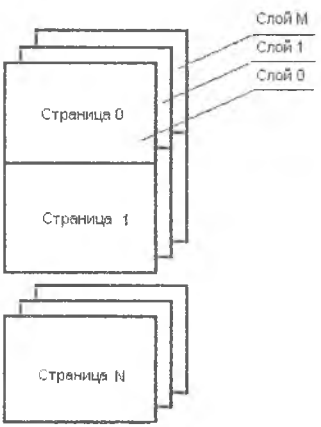


Рис. 3. Общая структура основной видеопамяти

Служебная память используется для внутренних операций видеоконтроллера, например, для хранения временных данных.

*Видео*BIOS - защищенный от записи фрагмент памяти с адресами 0C000h:0 - E000h:FFFFh (см. табл. 1), который содержит процедуры для управления видеоконтроллером. Они программно доступны через прерывание 10h.

Схема декодирования конвертирует информацию об изображении в числа, пропорциональные управляющим напряжениям. *ЦАП* конвертирует числа в управляющие напряжения для ЭЛТ.

Таблица 1

Типичное распределение памяти ПЭВМ

С адреса	По адрес	Назначение
0:0	0:3FFh	Вектора прерываний
40h:0	40h:FFh	Временные переменные BIOS и видеоBIOS
50h:0h	???	Ядро DOS
???	9000h:FFFFh	Свободная память для программ
A000h:0	B000h:FFFFh	Основная видеопамять
C000h:0	E000h:FFFFh	ВидеоBIOS (прерывание 10h)
F000h:0	F000h:FFFFh	BIOS (диск – 13h, COM-порты – 14h, клавиатура – 16h, и пр.)
10000h:0	10000h:FFFFh	HMA – high memory area
10000h:FFFFh	???	Расширенная память – для Windows

2. Типы видеоконтроллеров

В таблице 2 приведены основные типы видеоконтроллеров и даты их создания.

Таблица 2

Наиболее распространенные видеоконтроллеры

Год создания	Модель	Предельные возможности
1981 г.	MDA	2-хцветные алфавитно-цифровые изображения
1982 г.	CGA	4-цветные графические изображения
1984 г.	EGA	16-цветные графические изображения
1987 г.	VGA	256-цветные графические изображения
1991 г.	SVGA	24- и 32-битные графические изображения

Более новые модели видеоконтроллеров поддерживают режимы работы всех более ранних моделей.

3. Обзор видеорежимов

Видеорежим – это стандартизованный набор установок видеоконтроллера, позволяющий формировать изображение с определенными характеристиками. Обычно описывается как X×Y×C, где X - максимальное количество знаков или точек по горизонтали; Y - максимальное количество знаков или точек по вертикали; C - максимальное количество цветов.

3.1. Текстовые видеорежимы

Режимы 0 и 1. Алфавитно-цифровой режим 40x25x16.

При использовании видеоадаптеров EGA или VGA не существует функциональных различий между режимом 0 и режимом 1. В данных режимах дисплей отображает цветную текстовую (алфавитно-цифровую) информацию - 25 строк и 40 столбцов.

Для отображения каждого символа используется матрица 8 на 8 пикселов, что соответствует низкому качеству изображения (можно различить отдельные пиксели из которых состоит символ).

Символы текста можно отображать в 8 основных и 8 дополнительных цветах. Последние имеют большую интенсивность, чем основные. Для каждого символа можно независимо задать его цвет и цвет фона. Список стандартных и дополнительных цветов представлен в таблице 3.

Таблица 3

Стандартные и дополнительные цвета

Стандартный цвет	Дополнительный цвет
черный	серый
синий	светло-синий
зеленый	светло-зеленый
морской волны	голубой
красный	светло-красный
фиолетовый	малиновый
коричневый	желтый
белый	ярко-белый

Для видеоадаптеров EGA и VGA можно изменить используемую палитру цветов. EGA с улучшенным цветным дисплеем позволяет выбрать 16 цветов из 64 возможных, а VGA 16 из 262144.

В режимах 0 и 1 адаптеры EGA и VGA поддерживают восемь страниц видеопамяти. Страницей называется часть видеопамяти, полностью определяющая содержимое одного экрана дисплея. Одна из этих восьми страниц является активной, то есть ее содержимое отображается на экране. Для изменения активной страницы можно либо вызвать соответствующую функцию BIOS, либо непосредственно изменить содержимое регистра начального адреса, расположенного в контроллере электронно-лучевой трубки.

Режимы 2 и 3. Алфавитно-цифровой режим 80x25x16. Для видеоадаптеров EGA и VGA данные режимы не имеют различий. Это стандартный режим для MS-DOS.

Видеопамять имеет 1 слой и 8 страниц. По умолчанию активна страница 0, ее стартовый адрес B800h:0. Один знак на экране кодируется двумя соседними байтами:

- четный (0, 2, 4...) байт содержит числовой код знака;
- нечетный (1, 3, 5...) байт содержит описание цвета знака и цвета фона.

Формат байта атрибутов:

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
B3	*B2*	*B1*	*B0*	*S3*	*S2*	*S1*	*S0*

Биты S0-S3 кодируют один из 16 цветов знака (0 – черный, 1-синий,... 7-светло-серый, 8-темно-серый, 9-голубой, ... 15-белый). Биты B0-B2 кодируют один из 8 цветов фона. Бит B3 либо является дополнительным битом цвета фона (тогда фон может выбираться не из 8, а из 16 цветов), либо кодирует признак мерцания знака (1-мерцает, 0-нет). Режим бита B3 переключается с цвета на мерцание функцией видеоBIOS с номером 1003h:

```
r.ax:=$1003;
r.bl:=0-цвет;1-мигание.
Intr($10,r);
```

На самом деле изображение знака **аппаратно** (при помощи *знакогенератора*) моделируется матрицей светящихся точек, на CGA: 8x8, на EGA 8x14, на VGA 8x16 или 9x16. ВидеоBIOS хранит внутри таблицы с описаниями формы всех знаков (без русских букв!). Эти таблицы могут быть перепрограммированы (см. п. 4.1).

3.2. Графические видеорежимы

Режим 4 и 5. Графический режим 320x200x4. При отображении могут использоваться либо четыре основных, либо четыре альтернативных цвета: 1) черный, зеленый, красный, желтый; 2) черный, голубой, малиновый, белый.

Четверка цветов может выбираться при помощи функции 0Bh видеоBIOS. Ниже приведен фрагмент кода, позволяющий изменить четверку цветов:

```
r.ah:=$0B;
r.bh:=1;
r.bl:=номерцветовогонабора,0или1
Intr($10,r);
```

Видеопамять в режиме 4 имеет 1 слой и 1 страницу, разделенную на две части. Первая часть начинается по адресу 0B800h:0 и содержит описания четных (0, 2, 4...) строк, вторая – с адреса 0B800:2000h и содержит описания нечетных (1, 3, 5...) строк изображения. Цвет каждой точки кодируется двумя соседними битами (рис. 4).

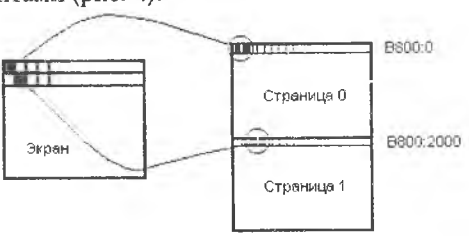


Рис. 4. Видеопамять в режиме 4

Формулы для расчета адреса пиксела в видеопамяти:

Nбайта=50h((y-1)/2)+(x/4)* – для четных строк;

Nбайта=Nбайта+2000h – для нечетных строк;

*Nбита=7-(xmod4)*2.*

Режим 6. Графический 640x200x2. Видеопамять начинается с адреса B800h:0, имеет 1 слой и 1 страницу, разделенную на две части для четных и нечетных строк (см. режим № 4). Два цвета (черный или белый) кодируются одним битом.

Формулы для расчета:

Nбайта=50h(y/2)+(x/8)* – для четных строк;

Nбайта=Nбайта+2000h – для нечетных строк;

Nбита=7-(xmod8).

Режим 12h. Графический 640x480x16 – режим, который устанавливается в MS Windows по умолчанию. Видеопамять начинается с адреса A000h:0, имеет одну страницу и 4 слоя. Цвет кодируется 4-мя битами, расположенными под одним и тем же номером в одном и том же байте, но в разных слоях.

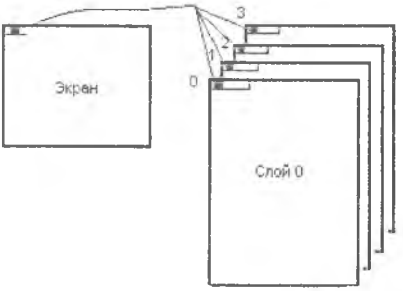


Рис. 5. Видеопамять в режиме 12h

Формулы для расчета:

```
Nбайта=50h*y+x/8;
Nбита=7-(xmod8).
```

Для того, чтобы поставить точку на экране, необходимо:

1) прочитать 4 байта из 4 разных слоев;
2) исправить нужные биты;
3) записать 4 байта на прежние места.

Внимание: при этом используются внутренние регистры контроллера (см. ниже п. 4.2.4 и п. 4.2.5).

Режим 0Fh. Графический режим 640x350x4. Видеопамять начинается с адреса A000h:0, имеет одну страницу и 2 слоя. Цвет кодируется 2-мя битами, расположенными под одним и тем же номером в одном и том же байте, но в разных слоях (см. режим № 12h).

Формулы для расчета:

```
Nбайта=50h*y+x/8;
Nбита=7-(xmod8).
```

Режим 13h. Графический режим 320x200x256. Видеопамять начинается с адреса A000h:0, имеет 1 слой и 1 видеостраницу. Одной точке изображения соответствует целиком байт видеопамяти, цвет в диапазоне 0-256 задается всеми восемью битами.

Формула для расчета:

```
Nбайта=140h*y+x.
```

Пример:

```
(*отобразить пиксел с координатами (123; 45) и
заданнымцветомврежиме13h*)
x:=123;(*КоординатаX*)
y:=45;(*КоординатаY*)
Mem[$A000:$140*y+x]:=цвет;
```

Режим 320x400x256. Видеоадаптер VGA технически способен работать в этом видеорежиме, но он не является стандартным и не поддерживается процедурами видеоBIOS.

3.3. VESA-расширения

В первой половине 90-х годов была попытка стандартизовать SVGA (проект VESA). Были предложены следующие видеорежимы (здесь указаны только некоторые из них):

Режим №100h: графический 640x400x256;
Режим №101h: графический 640x480x256;
...
Режим №103h: графический 800x600x256;
...
Режим №105h: графический 1024x768x256;
...
Режим №108h: текстовый 80x60x16;
Режим №109h: текстовый 132x25x16;
...
Режим №10Ch: текстовый 132x60x16;
...
Режим №118h: графический 1280x1024x32 бита;

Был разработан бесплатный универсальный драйвер для MS-DOS под названием UNIVBE (его можно найти на лазерных дисках со старыми игрушками), который для старых и несовместимых видеоконтроллеров моделировал необходимые процедуры видеоBIOS. Но стандарт так и не был закреплен документально, поэтому до сих пор в MS Windows для поддержки этих видеорежимов используются сотни различных драйверов.

Как проверить, поддерживает ли видеоконтроллер соглашения VESA:

```
r.ah:=$4F;
Intr($10,r);
If(r.al=$4F)thenwriteln('Yeah!');
elsewriteln('Damn!');
```

4. Программирование видеоконтроллеров

Видеоконтроллеры можно программировать:

- средствами видеоBIOS;
- непосредственно, через регистры контроллера.

4.1. Программирование средствами видеоBIOS

ВидеоBIOS предоставляет десятки сервисных функций для программирования видеоконтроллера в стандартных видеорежимах (с 1 по 13h). Функции доступны через прерывание с номером 10h, поэтому **использовать их можно только в MS-DOS программах**.

Недостатки:

- это очень медленно работающие функции;
- невозможно использование нестандартных возможностей видеоконтроллера;
- часть функций стандартизована, а часть может просто отсутствовать (например, VESA-функции).

Поэтому функции видеоBIOS практически никогда не используются для создания компьютерных игр, демонстрационных программ и других графических приложений.

1. Установка видеорежима.

```
r.ah:=0;
r.al:=номеррежима
Intr($10,r);
```

2. Определение текущего видеорежима.

```
r.ah:=$F;
Intr($10,r);
Writeln('Видеорежим□',r.al);
```

3. Рисование точки (**использовать в лабораторных работах запрещено**!).

```
r.ah:=$Ch;
r.al:=цвет
r.bh:=номерстраницывидеопамяти(обычно0)
r.cx:=X;
r.dx:=Y;
Intr($10,r);
```

4. Определение цвета точки (**использовать в лабораторных работах запрещено**!).

```
r.ah:=$D;
r.bh:=номерстраницывидеопамяти
r.cx:=X;
r.cy:=Y;
Intr($10,r);
Writeln('Цветточки=',r.al);
```

5. Получение информации о таблицах алфавитно-цифровых знаков (символов).

Эти шрифты используются в стандартных текстовых и графических режимах (**не в Windows !!!**). Изображения знаков хранятся поточечно, каждой точке соответствует один бит (см. рис. 6).

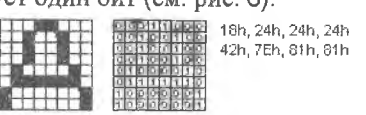


Рис. 6. Кодирование знаков 8x8

```
r.ax:=$1130;
r.bh:=Видзапроса(см.таблицу3)
Intr($10,r);
Writeln('Высотазнакавточках=',r.cl);
Writeln('Умещаетсянаэкранестрок=',r.dl+1);
Writeln('Адрес в памяти для таблицы описания знаков
=',r.es,':',r.bp);
```

Таблица 4

Кодирование запросов

Код запроса	Какая информация запрашивается
2	Информация про шрифт 8x14
3	Информация про шрифт 8x8 (1-я половина)
4	Информация про шрифт 8x8 (2-я половина)
5	Информация про шрифт 9x14
6	Информация про шрифт 8x16
7	Информация про шрифт 9x16

6. Установка своей таблицы знаков.

Можно поменять стандартные изображения знаков на свои (так поступают программы-русификаторы в MS-DOS, поскольку по умолчанию в видеоBIOS нет описания изображений русских букв).

```
r.ax:=$1100;
r.es:=Seg(Table);адрестаблицысновымнабором
r.bp:=Ofs(Table);символов
r.cx:=числозагружаемыхсимволов(1-256)
r.dx:=относительноесмещениепервогоизменяемого
символа(0-255)
r.bl:=номерзагружаемойтаблицызнакогенератора
дляEGA0-3,дляVGA0-7;
r.bh:=числобайтотводимыхнасимволвтаблице
символов(1-32);
```

4.2. Низкоуровневое программирование видеоконтроллера

Видеоконтроллер содержит несколько сотен регистров, разделенных на группы. Регистры программно доступны через порты ввода вывода при помощи машинных команд IN и OUT. С точки зрения доступа имеется два больших класса регистров:

- внешние регистры;
- внутренние регистры.

Внешние регистры называются так потому, что в видеоадаптере EGA они не принадлежат центральной микросхеме, содержащей контроллер атрибутов, контроллер ЭЛТ, графический контроллер и преобразователь последовательности. В адаптере VGA все регистры находятся на одной микросхеме, но эти регистры традиционно называют "внешними". В отличие от остальных регистров, внешние регистры адресуются непосредственно по адресам своих портов, без использования индексного регистра. Например, регистр состояния ввода № 1 всегда доступен через порт 3DAh.

Внутренние регистры по назначению и использованию разбиты на группы. С каждой группой связаны два порта: индексный порт и порт данных. В *индексном порту* задается номер интересующего регистра (0, 1, 2 и т.д.). Через *порт данных* ведется обмен информацией с указанным регистром.

4.2.1. *Некоторые внешние регистры*

*Регистр состояния ввода №*1. Адрес порта ввода-вывода 3DAh.

7	6	5	4	3	2	1	0
				X			

Если значение бита 3 равно 0, то идет прямой ход луча; если 1 – то обратный ход. Если изменять изображение во время прямого хода луча, то возможно подергивание и искажение экрана, поэтому в компьютерных играх запись в видеопамять обычно выполняется во время обратного хода.

4.2.2. *Некоторые внутренние регистры ЦАП*

Индексный порт чтения палитры – 3C7h.

Индексный порт записи палитры – 3C8h.

Порт данных для доступа к палитре – 3C9h.

В видеоконтроллере хранится *палитра* (или *таблица цветов*) – таблица соответствия номера цвета и конкретного значения. При включении компьютера эта таблица инициализируется стандартными значениями: 0 – черный, 1 – синий, 2 – зеленый… Эта таблица цветов (палитра) может быть перепрограммирована!

Для режима 13h (320x200x256) палитра представляет собой массив из 256 строк (по количеству цветов), каждая из строк состоит из 3 байтов, каждый из этих байтов содержит компоненту R, G или B (см. рис. 7).

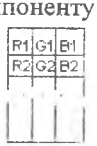


Рис. 7. Структура 256-цветной палитры

Реально в каждом байте используются 6 битов, т.е. каждая составляющая может принимать значения от 0 до 63. Таким образом, всего возможно 64x64x64=2^{18}=262144 различных цветов, из которых на экране одновременно может присутствовать только 256.

Как прочитать текущую палитру:

```
TypeRGB=recordR,G,B:byteend;
VarPal:array[0..255]ofRGB;
...
Port[$3C7]:=0;
(*Триподрядчтенияизодногопортадаютдоступкразным
регистрам*)
(*После тройного чтения индексный регистр
автоматическиинкрементируется*)
Fori:=0to255dobegin
Pal[i].R:=port[$3C9];
Pal[i].G:=port[$3C9];
Pal[i].B:=port[$3C9];
End;
```

Записать свой вариант палитры можно аналогично чтению, только в роли индексного порта используется 3C8h.

4.2.3. *Некоторые внутренние регистры контроллера ЭЛТ*

Индексный порт – 3D4h.

Порт данных –3D5h.

Эта группа регистров определяет формат данных в видеопамяти для изображения точек, знаков, курсора и т.п.

1. *Индекс* 8: регистр начальной линии изображения. Актуален и в графических, и в текстовых режимах. Позволяет в текстовых режимах сдвигать изображение не на целую строку, а на часть ее (см. рис. 8).

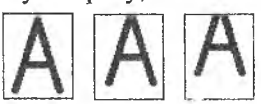


Рис. 8. Сдвиг букв на часть высоты строки

В регистре используется 7 младших битов, задают номер линии развертки для верхнего края экрана.

```
Fori:=0to13dobegin
Port[$3D4]:=8;
Port[$3D5]:=i;
End;
```

2. *Индекс* 9: высота символов текста в линиях (в точках) – 1, используется 5 битов.

3. *Индекс* 12h: высота экрана в линиях (в точках) – 1. Позволяет изменять размер видимой части экрана. Для видеоадаптера EGA регистр завершения отображения вертикальной развертки (VDER) содержит 9, а для VGA - 10 бит. Девятый и десятый биты находятся в дополнительном регистре (OVR), доступном по индексу 7. Причем девятый бит регистра VDER находится в бите D1, а десятый – в бите D6 дополнительного регистра OVR.

Пример. Нестандартный текстовый видеорежим 80x50x16. На VGA/SVGA по умолчанию используется 25 строк с символами высотой 16 точек, следовательно, высота экрана = 25x16=400 точек. Если задать высоту шрифта 8 точек, то на экране уместится 400/8=50 строк. Используем вышеописанные регистры:

```
Число_строк:=50;
{Задаемновуювысотусимволов}
Port[$3D4]:=9;Port[$3D5]:=7;
{Задаемновуювысотуэкрана}
Port[$3D4]:=$12;  Port[$3D5]:=Число_строк*8-256-
1;
{Вдоп.регистрзаносимстаршиебитычисла399}
Port[$3D4]:=7;OVR:=Port[$3D5];
OVR:=OVRor$02;
Port[$3D4]:=7;Port[$3D5]:=OVR;
```

```
{Загружаемшрифт8x8}
r.ax:=$1123;r.bl:=3;Intr($10,r);
{ИнформируемвидеоBIOSоновыхпараметрах}
mem[$40:$84]:=Число_строк-1;
```

4.2.4. *Некоторые внутренние регистры синхронизатора*

Индексный порт – 3C4h.

Порт данных – 3C5h.

Эта группа регистров управляет временными параметрами работы видеоконтроллера и доступом к отдельным цветовым слоям.

Индекс 2: регистр разрешения записи цветового слоя. **Применяется для рисования в многослойных режимах** (например, в 12h). Используются 4 младших бита.

7	6	5	4	3	2	1	0
				X	X	X	X

Каждый бит отвечает за свой цветовой слой: если = 1, то при записи в видеопамять данные попадают в этот слой; если = 0, то нет. Если все 4 бита установлены в 1, и мы записываем по какому-то адресу видеопамяти байт, то этот байт запишется сразу во все слои. Чтобы получать цвета, отличные от 0000 или 1111, надо поочередно разрешать-запрещать разные слои и писать в них разные значения.

4.2.5. *Внутренние регистры графического контроллера*

Индексный порт – 3CEh.

Порт данных – 3CFh.

Эта группа регистров используется для преобразования данных при обмене с видеопамятью.

Индекс 4: регистр разрешения чтения цветового слоя. Применяется для рисования в многослойных режимах (например, в 12h). Два младших бита используются для задания номера читаемого цветового слоя.

7	6	5	4	3	2	1	0
						X	X

Пример. Требуется в видеорежиме 12h поставить точку красного цвета (в стандартной палитре код=4) с координатами x=9, y=2. По формулам из п.3.2 получаем: Nбайта=50h*2 + 9/8 = 161; Nбита=7-(9 mod 8)=6.

```
{Прочитатьисохранитьданныеизвсехслоевдляданного
адресавидеопамяти}
  port[$3CE]:=4;
  port[$3Cf]:=0;{Разрешитьдлячтения0-ойслой}
  B[0]:=mem[$A000:161];
  port[$3CE]:=4;
  port[$3Cf]:=1;{Разрешитьдлячтения1-ыйслой}
  B[1]:=mem[$A000:161];
```

```
port[$3CE]:=4;
port[$3Cf]:=2;{Разрешитьдлячтения2-ойслой}
B[2]:=mem[$A000:161];
port[$3CE]:=4;
port[$3Cf]:=3;{Разрешитьдлячтения3-ийслой}
B[3]:=mem[$A000:161];

{Записатьвслоидля6-тобитакодкрасногоцвета=4}
B[0]:=B[0]and$BF;{x0xxxxxx}
B[1]:=B[1]and$BF;{x0xxxxxx}
B[2]:=B[2]or$40;{x1xxxxxx}
B[3]:=B[3]and$BF;{x0xxxxxx}

{Загрузитьмодифицированныеданныеввидеопамять}
port[$3C4]:=2;
port[$3C5]:=1;{Разрешитьдлязаписи0-ойслой}
mem[$A000:161]:=B[0];
port[$3C4]:=2;
port[$3C5]:=2;{Разрешитьдлязаписи1-ыйслой}
mem[$A000:161]:=B[1];
port[$3C4]:=2;
port[$3C5]:=4;{Разрешитьдлязаписи2-ойслой}
mem[$A000:161]:=B[2];
port[$3C4]:=2;
port[$3C5]:=8;{Разрешитьдлязаписи3-ийслой}
mem[$A000:161]:=B[3];
```

5. Некоторые методы генерации изображений

5.1. Клеточные автоматы

Клеточным автоматом называется математический объект, состоящий из:

- множества ячеек (например, это может быть квадратная матрица), каждому элементу которого соответствует числовое или логическое значение;
- системы правил, задающих рекуррентные преобразования значений в ячейках.

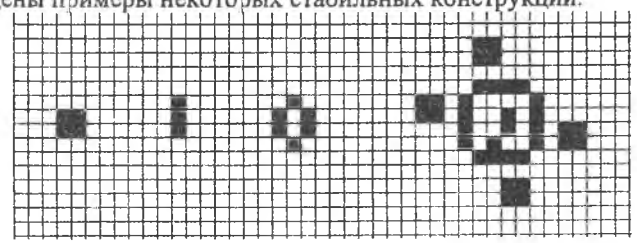
Обычно элементы матрицы отождествляются с точками растрового изображения, их числовые значения - с цветом точки, а рекуррентные преобразования – с процессом динамической визуализации. В терминах теории графов множество элементов, "соседних" с определенным элементом, называется *множеством Мура*. Например, для каждого из элементов a_{ij}, расположенных на плоскости в виде прямоугольной матрицы, множество Мура $M(a_{ij})$ состоит из 8 элементов: $a_{i-1,j-1}$, $a_{i,j-1}$, $a_{i+1,j-1}$, $a_{i+1,j}$ и т.д. Рекуррентные преобразования всех элементов клеточного автомата выполняются **одновременно**.

"Жизнь" Конвея

Элементами матрицы являются логические значения "истина" или "ложь". Правила преобразования:

- если значение элемента матрицы есть "истина", а суммарное число "истинных" элементов во множестве Мура больше трех или меньше двух, то элементу матрицы присваивается значение "ложь";
- если значение элемента матрицы есть "ложь", а суммарное число "истин" во множестве Мура равно трем, то элементу матрицы присваивается значение "истина".

"Жизнь" моделирует развитие колоний микроорганизмов, рост кристаллов и т.п. Большинство конструкций являются нестабильными и склонны либо к исчезновению, либо к неограниченному разрастанию. На рис. 9 приведены примеры некоторых стабильных конструкций.



а) "ящик" б) "мигалка" в) "колодец" г) "часы"

Рис. 9. Стабильные конструкции конвеевской "Жизни"

5.2. Генерация "Пламени"

Клеточный автомат "Пламя" (см. рис. 10, а) часто применяется при цифровом синтезе динамических изображений (например, в видеоклипах). Элементами матрицы являются числовые значения на интервале [0..MAX], отождествляемые с градациями цветов пламени: от черного, через красный и желтый к белому (это достигается перепрограммированием таблицы цветов видеоадаптера), см. рис. 10, б.

а) "пламя" б) варианты "палитры"

Рис. 10. Генерация "Пламени"

Шаг 1. Строится и загружается градиентная палитра (т.е. палитра с плавными переходами цветов) вида «белый- желтый- оранжевый- красный- черный».

Шаг 2. В нижних строках матрицы формируются "затравки" - наборы элементов с какими-то (например, случайными) значениями.

Шаг 3. В бесконечном цикле для каждой точки вычисляется среднее арифметическое цветов соседних точек, и это значение помещается на одну строку выше текущей.

Ниже приведен код, реализующий один из множества вариантов построения градиентной палитры.

```
{записьвпалитру}
ProcedureSetRGB(NumColor,r,g,b:byte);
Begin
port[$03C8]:=NumColor;
port[$03C9]:=R;
port[$03C9]:=G;
port[$03C9]:=B;
End;
{созданиеградиентнойпалитры}
ProcedureSetPalette;
Var x:byte;
Begin
forx:=0to15doSetRGB(x,0,0,0);
forx:=16to47doSetRGB(x,((x-16)shl1),0,0);
forx:=48to79doSetRGB(x,63,(x-48)shl1,0);
forx:=80to143doSetRGB(x,63,63,x-80);
forx:=143to255doSetRGB(x,63,63,63);
End;
```

5.3. Генерация "Плазмы"

Клеточный автомат "Плазма" (см. рис. 11) часто применяется при цифровом синтезе динамических изображений (например, в видеоклипах). Элементами матрицы являются числовые значения на интервале [0..MAX], отождествляемые с градациями цветов градиентной палитры (например, "радужной").

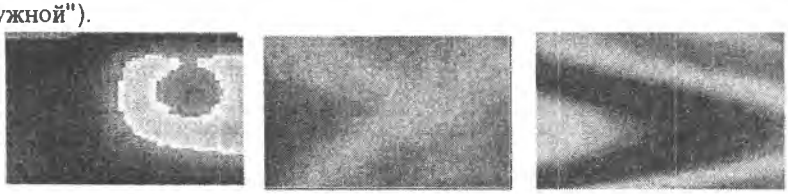


Рис. 11. Генерация "Плазмы"

Шаг 1. На экране формируются цветовые области с плавными переходами цветов от одного пятна к другому. Используются два способа:

- несколько закрашенных случайным цветом геометрических фигур со случайным местоположением и случайным размером подвергаются "размазыванию" при помощи метода, аналогичного использованному при генерации "Пламени";
- экран разбивается на прямоугольные области небольшого размера, в углах областей ставятся цветовые точки со случайными значениями, затем каждый прямоугольник рекурсивно закрашивается отрезками прямых по

правилу - цвет точки, лежащей на середине отрезка, равен среднему арифметическому между цветами его крайних точек.

Шаг 2. Выполняется "анимация" изображения. Используются два способа:

- производится "вращение палитры", т.е. циклический инкремент значений в таблице цветов;
- производится "вращение цветов точек", т.е. циклический инкремент цветовых значений пикселов.

5.4. Гашение/прояснение (fade/unfade) экрана

Заключается в плавном циклическом уменьшении/восстановлении компонент R, G, B палитры.

5.5. Фракталы

Фрактал – это объект сложной природы, получаемый в результате выполнения простого цикла итераций. Итерационность и рекурсивность обуславливают такое свойство фракталов, как самоподобие - отдельные части похожи по форме на весь фрактал в целом.

5.5.1. *Геометрические фракталы*

Геометрические фракталы строятся в результате применения к начальному множеству точек (к геометрической фигуре) последовательных преобразований по определенным правилам. Обычно такие преобразования выполняются рекурсивными алгоритмами до тех пор, пока минимальный элемент изображения превышает по величине один пиксел.

Пример 1. "*Губка Серпинского*"

Рис. 12. "Губка Серпинского"

Пример 2. "*Снежинка Коха*"

Рис. 13. "Снежинка Коха"

Пример 3. "*Дракон Хэтуэя*"

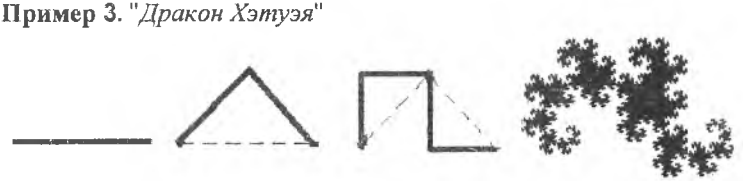


Рис. 14. "Дракон Хэтуэя"

Пример 4. "*Ковер Серпинского*"

Размер центрального прямоугольника определяют числа a и b, обозначающие отношение длины и ширины прямоугольника к общей величине области построения. Числа a и b находятся пределах от 0 до 1.

Ниже приведена структура рекурсивной процедуры выводящей ковер Серпинского.

```
ProcedureKover(x1,y1,x2,y2,a,b:byte);
begin
Ifразмер>=1пикселу
begin
Рисуемсреднийпрямоугольник;
Kover(правыйверхнийугол);
Kover(правыйнижнийугол);
Kover(левыйверхнийугол);
Kover(левыйнижнийугол);
end;
elseвыход;
end(Kover);
```

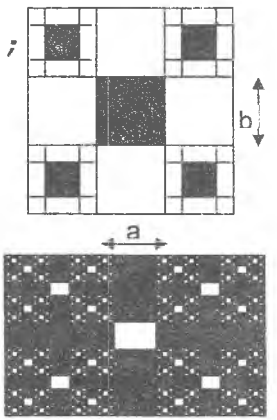


Рис. 15. Ковер Серпинского

5.5.2. *Аналитические фракталы*

Аналитические фракталы - реализации целочисленных функций комплексного аргумента n=f(z), где z=x+iy. Обычно x и y интерпретируются как координаты точки, а n - как ее цвет.

Схема вычисления функций имеет следующий вид:

```
n:=0;z0:=z;
while(УсловиеПродолжения(zn)&&(n<MAXCOLOR))
dobegin
zn+1:=ПравилоПреобразования(zn);
inc(n);
end;
```

Для создания фрактала, необходимо для каждой точки изображения выполнить цикл итераций согласно правилу преобразования. Стартовое значение $Z_0=x_0+iy_0$, где x_0, y_0 - координаты точки изображения, для которой выполняется цикл. Цикл выполняется ограниченное число раз (не более n) или до тех пор, пока не выполнится условие завершения цикла.

Пример 1. *Множество Жулиа* (Julia set) описывается формулой $Z_{k+1} = Z_k^2 + C$, где $C=C_x+iC_y$ - комплексная константа.

Условие остановки цикла $|Z_k| > 2$. Область построения изображения $x\in(-1; 1)$, $y\in(-1{,}2; 1{,}2)$. Значение комплексной константы C рекомендуется выбирать близким к (0,36+i·0,36).

Рис. 16. Множество Жулиа

Пример 2. *Множество Мандельброта* (Mandelbrot set) описывается формулой $Z_{k+1} = Z_k^2 + Z_0$. Область построения изображения $x \in (-2{,}2;\ 1)$, $y \in (-1{,}2;\ 1{,}2)$. Условие остановки цикла $|Z_k| > 2$.

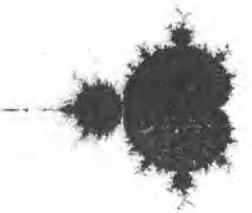


Рис. 17. Множество Мандельброта

Пример 3. *Крест Ньютона* описывается формулой $Z_{k+1} = \frac{3Z_k^4 + 1}{4Z_k^3}$. Область построения изображения $x \in (-1;\ 1)$, $y \in (-1;\ 1)$. Условие остановки цикла $\left|Z_k^4 - 1\right|^2 < \varepsilon$.

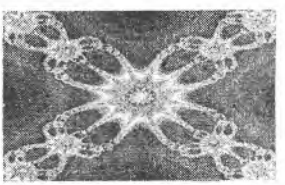


Рис. 18. Крест Ньютона

Арифметические действия над комплексными числами:

$|Z| = \sqrt{x^2 + y^2}$ - модуль комплексного числа;

$Z^2 = (x+iy)(x+iy) = x^2 - y^2 + i2xy$ - квадрат комплексного числа;

$(a+ib)+(c+id) = (a+c)+i(b+d)$ – сумма комплексных чисел;

$(a+ib)\cdot(c+id) = (a\cdot c-b\cdot d)+i(a\cdot d+c\cdot b)$ - произведение комплексных чисел;

$(a+ib)/(c+id) = (ac+bd)/(c^2+d^2)+i((cb-ad)/(c^2+d^2))$ – частное.

5.6. Спираль Станислава Улама

Целые числа располагаются по спирали, простые числа заменяются черными точками, остальные белыми.

а) нумерация точек плоскости б) расположение простых чисел

Рис. 19. Иллюстрация спирали Улама

Индивидуальные задания по теме "Использование технических возможностей графических контроллеров VGA/SVGA".

1. "Пламя" в режиме 320x200x256.
2. "Плазма" в режиме 320x200x256 (1 способ заливки).
3. "Плазма" в режиме 320x200x256 (2 способ заливки).
4. Плавное гашение/восстановление экрана в режиме 80x25x16.
5. Комментарии шрифтами 8x8, 8x14 и 8x16 одновременно в режиме 320x200x256.
6. Вращение символов в режиме 80x25x16.
7. Загрузить в таблицу знакогенератора символы: "звездочка", "сердечко", "снежинка" шрифтом 8x8 и вывести их в текстовом режиме.
8. "Снежинка Коха" в режимах 320x200x256 и 640x350x2.
9. "Снежинка Коха" в режимах 640x480x16 и 320x200x256.
10. "Пламя" в текстовом режиме 80x50x16 (или 80x43x16).
11. Одновременная демонстрация всевозможных цветов в режиме 320x200x256.
12. Вертикальный сдвиг на целое число пикселов в 80x25x16.
13. Рисование нестандартными символами в режиме 80x25x16.
14. "Улитка Джулии" в режимах 320x200x256 и 640x350x2.
15. "Улитка Джулии" в режимах 640x480x16 и 320x200x256.
16. "Крест Ньютона" в режимах 320x200x256 и 640x200x2.
17. "Крест Ньютона" в режимах 640x480x16 и 320x200x256.
18. Дрожание экрана с разными частотами.
19. "Фрактал Мандельброта" в режимах 320x200x4 и 320x200x256.
20. "Фрактал Мандельброта" 320x200x256 и 640x480x16.
21. Варианты "пламени" в 320x200x256 в различных цветовых тональностях: бело-желто-красной, бело-сине-фиолетовой, бело-зеленой и т.п.
22. Варианты "плазмы" в 320x200x256 в различных цветовых тональностях: бело-желто-красной, бело-сине-фиолетовой, бело-зеленой и т.п.
23. "Дракон Хэтуэя" в режимах 320x200x4 и 640x350x2.
24. "Губка Серпинского" в режимах 320x200x4 и 640x350x2.
25. "Ковер Серпинского" в режимах 320x200x4 и 640x200x2.
26. "Ковер Серпинского" в режимах 640x480x16 и 320x200x256.
27. "Жизнь" Конвея в режимах 640x200x2 и 640x350x2.
28. "Жизнь" Конвея в режимах 640x480x16 и 320x200x256.
29. Бегущая строка в текстовых режимах 1 и 3.
30. Бегущая строка в графических режимах 12h и 13h.
31. Построение спирали Станислава Улама в режимах 640x480x16 и 320x200x256.
32. Написать программу, рисующую на экране разноцветные квадраты различного размера. Предусмотреть возможность складывать цвета наложенных друг на друга квадратов. Проверить с помощью этой программы аддитивную цветовую систему (например, красный+синий=пурпур). Исследовать эффект одновременного контраста,

изображая маленький пурпурный квадрат внутри ярко-красного и ярко-синего.

33. Вывод на экран стереокартинки. Экран заполняется множеством маленьких черно-белых квадратиков, создающих хаос. Затем небольшая область экрана размером 50x50 пикселов считывается в массив и последовательно копируется восемь раз, в две строки по четыре квадрата впритык. Предусмотреть возможность перерисовки экрана по нажатию клавиши <пробел> для реализации возможности подобрать наиболее удачную картинку.
34. Построение графика функции поточечно. $f(x)=e^{-x}\cos(2\pi x)$.
35. Визуальная оценка качества генератора случайных чисел (0...N-1). Построить график рассеяния, используя пары последовательных случайных величин. Протестировать встроенный датчик, а также генератор псевдослучайных чисел, построенный по формуле $n_i=[n_{i-1} A+B] \bmod N$, где A = 1 103 515 245, B = 12 345, N = 32 767.

    Поэкспериментировать с выбором значений A, B и N.
36. Напишите программу, которая позволяет пользователю выбирать значения составляющих RGB цвета и выводит результирующий цвет в квадратной области экрана.
37. Тьюрмиты. Тьюрмит работает по программе. В каждый момент времени выполняется одна из строк программы. Какая – зависит от цвета клетки, в которой тьюрмит находится и от текущего состояния. Результатом выполнения этой строки будет следующее: тьюрмит покрасит клетку, на которой стоит, в заданный цвет, переместится в заданном направлении, узнает какую строку программы он должен выполнить следующей.

    Синтаксис строк программы тьюрмита: <состояние>,<текущий цвет>, <новый цвет>, <код перемещения>, <новое состояние>. Перемещений может быть только три: -1="налево", 0="прямо", 1="направо". Например, строка "A 0 15 1 A" означает, что после попадания на черное (цвет 0) поле находясь в состоянии A, тьюрмит перекрасит его в белый цвет (цвет 15), переместится направо и начнет искать строку в программе, начинающуюся буквой A и новым текущим цветом.

    Реализовать работу следующих программ:

Программа 1.	Программа 2.	Программма 3.
A 0 2 0 C	A 0 14 0 B	A 0 14 0 B
A 2 0 0 B	A 14 13 -1 A	B 0 14 0 C
B 2 2 1 A	A 13 0 1 B	C 0 14 0 E
B 15 2 1 A	B 0 13 1 A	E 0 14 0 F
C 2 0 -1 A	B 13 14 -1 A	F 0 14 0 J
C 0 15 -1 A	B 14 0 0 A	J 0 13 -1 A
C 15 2 -1 A		A 14 13 0 A
		A 13 14 0 A
		J 14 13 0 A
		J 13 14 1 A

Дополнительные вопросы по теме "Использование технических возможностей графических контроллеров VGA/SVGA"

1. Какова структура видеопамяти в текстовом режиме 3, сколько слоев и страниц?
2. Сколько бит видеопамяти используется для кодировки одного символа в текстовом режиме 3?
3. Какова структура видеопамяти в графических режимах 4h (320x200x4), 6h (640x200x2), 12h (640x480x16) и 13h (320x200x256)? Какое количество слоев и страниц используется в данных режимах?
4. Сколько бит видеопамяти используется для кодировки цвета пиксела в графических режимах 4h, 6h, 12h и 13h?
5. Сколько байт в таблице знакогенератора требуется для кодировки символа шрифта 8x8, 8x14, 9x16?
6. Для чего может использоваться функция отслеживания начала обратного хода луча ЭЛТ?
7. Какова структура палитры для режима 13h (320x200x256)?
8. Каким образом осуществляется доступ к отдельным цветовым слоям при рисовании в многослойных режимах (например, в 12h)?
9. Какой должна быть последовательность действий для того, чтобы вывести на экран пиксел 5-го цвета в многослойном режиме 12h?

Лабораторная работа № 2

Тема: ***"Алгоритмы построения и преобразования изображений"***

Лабораторная работа предназначена для ознакомления с основными алгоритмами построения и преобразования растровых изображений, такими, как быстрые (инкрементные) алгоритмы построения геометрических фигур, закраска замкнутых контуров, построение различных видов проекций, применение аффинных преобразований, использование матричных фильтров и т.д.

Лабораторная работа выполняется в графическом режиме 320x200x256.

Индивидуальные задания по теме "Алгоритмы построения и преобразования изображений"

1. Вращение и масштабирование фрагмента штрихового изображения.
2. Вращение "проволочной" фигуры (линии которой построены при помощи алгоритма Брезенхама) в плоскостях, перпендикулярных плоскости экрана.
3. Вращение символов 8x8 в плоскости экрана относительно заданного центра.
4. Вращение "проволочной" фигуры в плоскости экрана.
5. Облёт вокруг проволочной фигуры по траектории "спираль".
6. Наложение полутоновой текстуры на треугольник и вращение в плоскости экрана.
7. Построение объемного "проволочного" куба при помощи алгоритма Брезенхама и вращение его в плоскостях, перпендикулярных плоскости экрана.
8. Наложение полутоновой текстуры на треугольник и вращение в плоскостях, перпендикулярных плоскости экрана.
9. Построение множества случайно расположенных окружностей и замкнутых четырехугольников, границы которых построены при помощи алгоритма Брезенхама и "заливка" получившихся замкнутых фигур при помощи рекурсивного алгоритма с "затравкой".
10. Реализовать алгоритм заполнения полигонов отрезками прямых.
11. Построение случайного лабиринта, линии которого построены при помощи алгоритма Брезенхама и обход его рекурсивным алгоритмом
12. Построение "фигур Лиссажу" и заливка каждого сегмента отдельным цветом рекурсивным алгоритмом с "затравкой".
13. Вращение "фигур Лиссажу" в плоскости экрана.
14. Построение кривых Безье произвольного порядка (до 4-го включительно) и вращение их в плоскости экрана.
15. Построение изображения "коридора", стены и потолок которого покрыты полутоновыми текстурами.
16. Построение звездного неба в результате нескольких итераций игры "Conway's the Life" и вращение центрального круглого сегмента.
17. "Лупа". Масштабирование круглого фрагмента штрихового изображения.

18. "Лупа". Масштабирование прямоугольного фрагмента полутонового изображения.
19. Трансформации небольшого полутонового изображения: масштабирование, перемещение, вращение в плоскости экрана, симметричное отражение.
20. "Спрайты": перемещение небольшого изображения поверх другого.
21. Вращение фрагмента полутонового изображения в плоскости, перпендикулярной плоскости экрана.
22. Суперэллипсоид. Неявная форма задания $(x^n+y^n)^{m/n}+z^m-1$. Параметрическая форма задания $(\cos^{2/m}(v)\cos^{2/n}(u),\ \cos^{2/m}(v)\sin^{2/n}(u),\ \sin^{2/m}(v))$. Промежуток изменения параметров $v\in[-\pi/2,\ \pi/2]$, $u\in[-\pi,\ \pi]$. Изменяя значения m и n, получить различные вариации формы тела.
23. Супертороид. Неявная форма задания $((x^n-y^n)^{1/n}-d)^m+z^m-1$. Параметрическая форма задания $((d+\cos^{2/m}(v))\cos^{2/n}(u),\ (d+\cos^{2/m}(v))\sin^{2/n}(u),\ \sin^{2/m}(v))$. Промежуток изменения параметров $v\in[-\pi,\ \pi)$, $u\in[-\pi,\ \pi)$. Изменяя значения m и n, получить различные вариации формы тела.
24. Перевод полутонового цветного изображения в "серую" гамму, загрязнение его случайно расположенных пятен и устранение загрязнений.
25. Перевод цветного полутонового изображения в "оттенки серого", переход из серой гаммы в монохромную, утончение линий и выделение связных контуров при помощи алгоритма Рутовица.
26. Перевод цветного полутонового изображения в "оттенки серого" и последовательные матричные операции: 1) размывка контуров; 2) увеличение резкости.
27. Перевод цветного полутонового изображения в "оттенки серого", придание рельефности изображению.
28. Написать программу для применения к изображению произвольных матричных фильтров, задаваемых пользователем.
29. "Летающие часы".
30. "Вечный двигатель". Произвольное перемещение по экрану 4-х точек, соединенных прямыми, построенными при помощи алгоритма Брезенхама.

Дополнительные вопросы по теме "Алгоритмы построения и преобразования изображений"

1. Какие способы построения прямой линии вы знаете?
2. В чем заключается основная особенность алгоритма Брезенхама для построения прямой линии?
3. Какова основная идея алгоритма Брезенхама для построения окружностей и эллипсов?
4. Опишите геометрический способ построения кривых Безье.
5. В чем заключается рекурсивный алгоритм с "затравкой" для закрашивания замкнутых контуров?
6. Опишите идею построчного алгоритма и метода заполнения полигонов для закрашивания замкнутых контуров.
7. Какие виды афинных преобразований вы знаете?
8. Каковы основные свойства афинных преобразований?
9. Какие виды проекций вы знаете?
10. При каком виде проекции все лучи проецирования расположены под углом 90° к плоскости проецирования?
11. При каком виде проекции лучи проецирования исходят из одной точки?
12. Какие афинные преобразования используются при преобразовании координат проекции в экранные координаты?
13. Какие методы описания формы поверхностей вы знаете?

Рекомендуемая литература

1. *Фролов А.В.*, *Фролов Г.В.* Программирование видеоадаптеров CGA, EGA и VGA. Том 3, М.: Диалог-МИФИ, 1992, 287 стр.
2. *Вельтмандер П.В.* Машинная графика.- В 3 кн. Кн. 1. Вводный курс компьютерной графики.-Новосибирск: Изд-во НГУ, 1997.
3. *Вельтмандер П.В.* Машинная графика.- В 3 кн. Кн. 2. Основные алгоритмы компьютерной графики.-Новосибирск: Изд-во НГУ, 1997.

Таблица соответствия наиболее важных команд на языках C и Pascal

Описание	C	Pascal

Структура контейнера регистров	`structREGPACK{`
	`unsignedr_ax,r_bx,r_cx,r_dx;`
	`unsignedr_bp,r_si,r_di;`
	`unsignedr_ds,r_es,r_flags}`
		`type`
		`Tregisters=record`
		`caseIntegerof`
		`0:(AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags:Word);`
		`1:(AL, AH, BL, BH, CL, CH, DL, DH: Byte);`
		`end;`

Вызов прерывания	`voidintr(intintr_num, structREGPACK*preg);`	`procedure Intr(IntrNo: Byte; var Regs: TRegisters);`
	Пример: установка графического видеорежима 13h	

	`structREGPACKreg;`
	`reg.r_ax=0x0;`
	`reg.r_ax+=0x13;`
	`intr(0x10,&reg);`
		`varRegs:TRegisters;`
		`begin`
		`Regs.AH:=$00;`
		`Regs.AL:=$13;`
		`Intr($10,Regs);`
		`end;`

Чтение слова по адресу seg:off	`intpeek(unsignedseg, unsignedoff)`	`MemW[Segment:Offset]`
	Пример: чтение цвета пиксела с координатами (x, y) в режиме 13h	
	`intcolor;` `color=peek(0xA000,0x140*y+x);`	`varB:Integer;` `B:=MemW[$A000:$140*y+x];`
Чтение байта по адресу seg:off	`charpeekb(unsignedseg, unsignedoff)`	`Mem[Segment:Offset]`
	Пример: чтение цвета пиксела с координатами (x, y) в режиме 13h	
	`charcolor;` `color=peekb(0xA000,0x140*y+x);`	`varB:Byte;` `B:=Mem[$A000:$140*y+x];`

ПРИЛОЖЕНИЕ 1

Описание	C	Pascal
Запись слова по адресу seg:off	`voidpoke(unsignedseg, unsignedoff, intvalue)`	`MemW[Segment:Offset]`
	Пример: окрашивание пиксела с координатами (x, y) в цвет color в режиме 13h	
	`intcolor=0x0016; poke(0xA000,0x140*y+x,color);`	`varcolor:Integer; color:=$0016; MemW[$A000:$140*y+x]:=color;`
Запись байта по адресу seg:off	`voidpokeb(unsignedseg, unsignedoff, charvalue)`	`Mem[Segment:Offset]`
	Пример: окрашивание пиксела с координатами (x, y) в цвет color в режиме 13h	
	`charcolor=0x16; pokeb(0xA000,0x140*y+x,color);`	`varcolor:Byte; color:=$16; Mem[$A000:$140*y+x]:=color;`
Чтение слова из порта	`intinport(intnumport);`	`PortW[address]`
Чтение байта из порта	`unsignedcharinportb(intnumport);`	`Port[address]`
	Пример: прочитать составляющие R, G, B 7-го цвета из 256 цветной палитры	
	`structcolor{unsignedcharR,G,B;}; structcolorpalette[256]; outportb(0x3C7,7); palette[7].R=inportb(0x3C9); palette[7].G=inportb(0x3C9); palette[7].B=inportb(0x3C9);`	`TypeRGB=recordR,G,B:byteend; VarPal:RGB; Port[$3C7]:=7; Pal.R:=Port[$3C9]; Pal.G:=Port[$3C9]; Pal.B:=Port[$3C9];`
Запись слова в порт	`voidoutport(intnumport, intvalue);`	`PortW[address]`
Запись байта в порт	`voidoutportb(intnumport, charvalue);`	`Port[address]`
	Пример: сдвиг экрана в текстовом режиме на 4 линии вверх	
	`outportb(0x3d4,8); outportb(0x3d5,0x04);`	`Port[$3D4]:=8; Port[$3D5]:=4;`

Описание формата BMP файла

По решению разработчиков формат bmp-файла не привязан к конкретной аппаратной платформе. Этот файл состоит из четырех частей: заголовка, информационного заголовка, таблицы цветов (палитры) и данных изображения. Если в файле хранится изображение с глубиной цвета 24 бита (16 млн. цветов), то таблица цветов может отсутствовать, однако в нашем, 256-цветном случае она есть. Структура каждой из частей файла, хранящего 256-цветное изображение, дана в таблице П2.1.

Таблица П2.1

Структура BMP файла

Имя	Длина	Смещение	Описание
Заголовок файла (BitMapFileHeader)			
Type	2	0	Сигнатура "BM"
Size	4	2	Размер файла
Reserved 1	2	6	Зарезервировано
Reserved 2	2	8	Зарезервировано
OffsetBits	4	10	Смещение изображения от начала файла
Информационный заголовок (BitMapInfoHeader)			
Size	4	14	Длина заголовка
Width	4	18	Ширина изображения, точки
Height	4	22	Высота изображения, точки
Planes	2	26	Число плоскостей
BitCount	2	28	Глубина цвета, бит на точку
Compression	4	30	Тип компрессии (0 - несжатое изображение)
SizeImage	4	34	Размер изображения, байт
XpelsPerMeter	4	38	Горизонтальное разрешение, точки на метр
YpelsPerMeter	4	42	Вертикальное разрешение, точки на метр
ColorsUsed	4	46	Число используемых цветов (0 - максимально возможное для данной глубины цвета)
ColorsImportant	4	50	Число основных цветов
Таблица цветов (палитра) (ColorTable)			
ColorTable	1024	54	256 элементов по 4 байта
Данные изображения (BitMap Array)			
Image	Size	1078	Изображение, записанное по строкам слева направо и снизу вверх

Заголовок файла начинается с *сигнатуры* «BM», а затем идет длина файла, выраженная в байтах. Следующие 4 байта зарезервированы для дальнейших расширений формата, а заканчивается этот заголовок *смещением* от начала файла до записанных в нем данных изображения. При 256 цветах это смещение составляет 1078.

Информационный заголовок начинается с собственной длины (она может изменяться, но для 256-цветного файла составляет 40 байт) и содержит размеры изображения, разрешение, характеристики представления цвета и другие параметры.

Ширина и высота изображения задаются в точках растра.

Количество плоскостей могло применяться в файлах, имеющих небольшую глубину цвета. При числе цветов 256 и больше оно всегда равно 1, поэтому сейчас это поле уже можно считать устаревшим, но для совместимости оно сохраняется.

Глубина цвета считается важнейшей характеристикой способа представления цвета в файле и измеряется в битах на точку.

Компрессия. В bmp-файлах обычно не используется, но поле в заголовке для нее

предусмотрено. Обычно она равна 0, и это означает, что изображение не сжато.

Размер изображения - количество байт памяти, требующихся для хранения этого изображения, не считая данных палитры.

Горизонтальное и вертикальное разрешения измеряются в точках растра на метр. Они особенно важны для сохранения масштаба отсканированных картинок. Изображения, созданные с помощью графических редакторов, как правило, имеют в этих полях нули.

Число цветов позволяет сократить размер таблицы палитры, если в изображении реально присутствует меньше цветов, чем это допускает выбранная глубина цвета. Однако на практике такие файлы почти не встречаются. Если число цветов принимает значение, максимально допустимое глубиной цвета, например 256 цветов при 8 битах, поле обнуляют.

Число основных цветов — идет с начала палитры, и его желательно выводить без искажений. Данное поле бывает важно тогда, когда максимальное число цветов дисплея было меньше, чем в палитре bmp-файла. При разработке формата, очевидно, принималось, что наиболее часто встречающиеся цвета будут располагаться в начале таблицы. Сейчас этого требования практически не придерживаются, т.е. цвета не упорядочиваются по частоте, с которой они встречаются в файле. Это очень важно, поскольку палитры двух разных файлов, даже составленных из одних и тех же цветов, содержали бы их (цвета) в разном порядке, что могло существенно осложнить одновременный вывод таких изображений на экран.

За информационным заголовком следует ***таблица цветов***, представляющая собой массив из 256 (по числу цветов) 4-байтовых полей. Каждое поле соответствует своему цвету в палитре, а три байта из четырех — компонентам синей, зеленой и красной составляющих для этого цвета. Последний, самый старший байт каждого поля зарезервирован и равен 0.

После таблицы цветов находятся ***данные изображения***, которое по строкам растра записано снизу вверх, а внутри строки — слева направо. Так как на некоторых платформах невозможно считать единицу данных, которая меньше 4 байт, длина каждой строки выровнена на границу в 4 байта, т. е. при длине строки, некратной четырем, она дополняется нулями. Это обстоятельство обязательно надо учитывать при считывании файла, хотя, возможно, лучше заранее позаботиться, чтобы горизонтальные размеры всех изображений были кратны 4.

Так как формат файла был разработан универсальным для различных платформ, то нет ничего удивительного в том, что цвета палитры хранятся в нем иначе, чем принято для VGA. Во время выполнения процедуры чтения производится необходимая перекодировка.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ
"КОМПЬЮТЕРНАЯ ГРАФИКА"
(лабораторные работы № 1 и 2)

Методические указания

Составители: *Климентьев Константин Евгеньевич,*
Кудрина Мария Александровна.

Издательство Самарского научного центра РАН
Лицензия на издательскую деятельность ЛР 040910 от 10.08.98 г.

Подписано в печать 22.11.2005 г. Формат 60x84 1/16.
Бумага офсетная. Печать оперативная.
Усл. печ. л. 2.
Тираж 100 экз. Заказ № 921.

Отпечатано в типографии АНО "Издательство СНЦ РАН"
443001, Самара, Студенческий пер., 3а