Компютърната графика се определя като методи и средства за преобразуване на данни в или от графично изображение. В смисъла на това определение КГ, се разделя на генерираща и аналитична.

Генериращата графика е съвкупност от методи и средства за генериране на графични изображения. Обработват се компютърно генерирани геометрични модели на обектите и техните графични изображения.

При анализа на изображенията се обработват готови фотографски изображения. Или може да се каже че даденото графично изображение се преобразува във вид разбираем от компютъра. КГ създава изображения на неграфична информация, която може да бъде координатна, картинни модели и др.

Tema 2.

Има няколко основни модели на изображение:

а) За текст: ASCII (8-битов) код: основен и горен (разширен) регистри; 28 кодови комбинации

Unicode (16-битов): включва ASCII като подмножество от всичките 216 кодови комбинации (... японска и корейска азбуки).

Режим за дисплейване и отпечатване - Текстов - с вариране на широчини и броя на редовете.

б) За графика

Пиксел - точка върху отпечатана страница или екрана.

Атрибути на пиксела (описание за яркостта и цвета на всеки пиксел - RGB).

Режим за дисплейване и отпечатване - точкова графика (bitmap), на всеки пиксел отговаря код : Изображението се описва като масив от пиксели (bitmap), като разделителна способност се определя се от размерите и броя на пикселите например за VGA екран от 640x480 пиксели (307200) изисква 38,4 KB данни в ч/б режим. При 256 цвята (8 бита за пиксел) се изискват 8x38,4 KB. Приложни програми за работа с битмап са например : PaintBrush, Aldus PhotoStyler. Растерната графика не е хардуерно независима. Тя зависи от разделителната способност. Тя генерира графичните изображения като последователност от графични точки (пиксели) подредени в редове и колони. Сканира се последователно ред след ред, като чрез управление на интензивността или цвета на всяка точка се изгражда изображението. Тук изображението може да бъде много реалистично когато става въпрос за пространствени обекти, като това се постига чрез цветно изображение на светлосенки.

Файловите формати за растерна графика са (битмап) :

TIFF (Tagged Image File Format) на Aldus и Microsoft. TIFF версия 5.0 има 8-битов код на сивото (256 степени), 24-битови цветни изображения и черно-бяло изображение. PCX, създаден за програми PC Paintbrush и Publisher’s Paintbrush, поддържа 8-битови степени на сивото и цветове.

Обектно-ориентирана графика

Изображението се описва векторно, още т.нар. обектно-ориентираната графика (отсечки, окръжности, многоъгълници и криви). Предимства на векторната графика - малки размери на файловете, хардуерна независимост и описание с помощта на процедурни езици за програмиране. И може да се опише във файл, който позволява изход при максимална разрешаваща способност на устройството.

Тема 3..

Най-общо дисплейните подсистеми се състоят от дисплеен процесор, дисплеен генератор и дисплеен пулт. От геометричният модел се генерира код за процесора, който се нарича дисплеен файл. Процесора е интерпретатор на дисплейният файл, като интерпретира не само инструкциите му но и управлява Д генератор и обработва състоянията на цялата подсистема. В дисплейният файл се записват не само инструкциите на Д Процесор, но и отделните параметри на графичните примитиви на картината (яркост, цвят, тип на линиите и др.). Дисплейният пулт се състои от монитор и интерактивни средства. ДГ изработва аналогови сигнали за да може с тях да работи електронно-лъчевата тръба на монитора.

Мониторите биват векторни или растерни. От своя страна векторните се делят на още два типа: с регенерация на дисплейната картина, и със запомняща тръба. Поради свойството на електронният лъч да се движи само по права линия, то за да се начертае дъга например, тя трябва да се разбие на малки вектори, които я изграждат. По същият начин се изобразяват и буквените символи и всички неправилни символи или криви. С разбиването на малко отсечки (вектори) се занимава дисплейният процесор.

Растерните монитори са основани на принципа на TV мониторите. При тях изобразяването на нещо става посредством обхождане на всички пиксели от монитора ред по ред. Като интензивността и цвета се определят за всеки пиксел поотделно. При по-старите видове дисплей, честотата на опресняване на картината е по-малка, и обхождането на монитора става на два етапа: първо се обхождат четните после нечетните редове от екрана тази система се нарича Interleast. Командите генерирани от дисплейният генератор се съхраняват в памет наречена кадров буфер, където в последствие се изпраща към видеодисплея. Качеството зависи от броя на битове, които се отделят за съхраняване на състоянието на един пиксел от системата. Можем да варираме от 2 до 32 бита, като съответно по-качественият образ съответно се заплаща с повече ресурси. Тук имаме и таблица на цветовете, използване за смяна на цветността на изображението без да се сменя съдържанието на кадровият буфер. Един пиксел се дели на три основни цвята. За да изобразим необходимият ни цвят, то трябва да зададем различна стойност на интензивност на светене на трите цвята.

ТЕМА 4:

1.K*оординатна система и текущ указател* (CP - Current Pointer). Координатната система на екрана, започва от горен ляв ъгъл, и нараства към долен десен ъгъл. В графичен режим имаме достъп и управление върху всеки пиксел от екрана. Можем да минем в графичен режим, чрез инициализация на системата.

2.*Инициализация на графичната система*. За да преминем в графичен режим, ни трябва специален драйвер за типа видеодисплей който използваме. В Паскал графичните драйвери имат специфични номера, които са показани в следващата таблица :

Current

Вид монитор

Номер

Current Driver

-128

Detect

0

CGA

1

MCGE

2

EGA

3

EGA64

4

EGAMono

5

Herc mono

6

IBM 8414

7

ATT 400

8

VGA

9

PC3270

10

При завършване на работа с графичната система, е необходимо тя да бъде затворена.

4.Графични адаптери - устройства, които управляват дисплея.

а) видеопамет - една част от паметта на компютъра, е отделена за запомняне на стойностите за атрибутите на пиксела (един пиксел се образува от три светещи с различен интензитет и цвят точки - RGB технология). Видеоадаптера сканира непрекъснато сканира непрекъснато видеопаметта, и преобразува стойностите на пикселите в сигнали за управление на електронният лъч.

б) Графични страници - една графична страница е част от видеопаметта съдържаща стойността на всички пиксели попадащи в нея. Трябва да разполагаме поне с две графични страници, за да може докато в едната се записват данните за пикселите от другата да се излъчва нещо на екрана.

в) Карта на цветовете - в тази карта се записват номерата на цветовете и техните вектори. Вектора на цвета може да има по-голяма стойност от номера на самият цвят. Посредством тази карта, адаптера преглежда на кой номер цвят, какъв вектор отговаря. Колкото повече битове се отделят, толкова по-голяма е картата на цветовете, и от там по-качествени са изображенията (и по-големи естествено).

г) Режими - режимите биват няколко вида:

графични режими - различават се по разделителната способност, броят на поддържаните цветове, и броят на графичните страници. В съвременните гр. Адаптери се поддържат няколко графични режима. Ето и таблица с няколко най-разпространени графични режими :

Име

разд. Сп.

bite/pix.

Color

бр. стр.

памет

адрес

CGA

320x200

2

4

1

16 k

B8000H

640x200

1

2

1

EGA

640x200

4

16

4

64 к

D0000H

640x350

4

16

2

VGA

640x480

4

16

1

256

A0000H

320x200

8

256

1

640x200

4

16

4

IBM851

640x480

256

A0000H

1024x768

512

Графични драйвери - програми, които управляват работата на дисплея заедно с графичният адаптер. Може да използваме за един графичен адаптер няколко графични драйвери, с цел да имитират работата на различни адаптери.

Тема 5:

При растерната графика, имаме определена големина на пиксела. Може да се случи така, че да имаме нужда един пиксел да свети наполовина, което е невъзможно. Ето защо се налага да използваме алгоритми за растеризация.

За да може да се изчертават по-качествени (реални) образи или по-близки до оригинала.

Целта на този алгоритъм е да намерим тези пиксели, които са възможно най-близки до идеалната права пиксели, образуващи даденият обект. Този алгоритъм има чисто математическо обяснение, в което се използват уравненията за права минаваща през две точки. Ако началната точка е с координати (X_1,Y_1) êðàéíàòà ñ êîîðäèíàòàòà (X_2,Y_2), уравнението на правата ще има вида: =>

;, $b=Y_1-mX_1$, $m=tg$.

Както знаем през две точки може да мине само една права. Нека в такъв случай имаме зададени двете крайни точки на една отсечка. То в такъв случай за да построим отсечката, трябва да прокараме през точките права, възможно най-близка до идеалната, по най-бързият и най-точният начин. Веднага изхождаме от математическата формула за построяване на уравнение на права по зададени две нейни точки.

Процедурата за изчертаване на правата по последният математически алгоритъм е следната:

```
Procedure LineDraw(X1, Y1, X2, Y2 : integer);
Var
X, DX, DY : integer;

m, b, Y, : real;

Begin

dX:=X2-X1;

dY:= Y2-Y1;

m:=dY/dX;

b:=Y1-m*X1;

X:=X1;
Repeat
```

```
Y:=m*X+b;


Putpixel(X, Round(Y), color);

  X:=X+1;


Until X>X2;


End;
```

DDA(Digital Differential Analyses) алгоритъмът е цифров защото се използва тангенса на ъгъл , където ако ъгълът е > 45^o се прави нарастване по Y и ако е по-малък по X. Този алгоритъм е в сила само за прави в първи квадрант, или когато 90> >0;

```
Procedure DDA(X1,Y1,X2,Y2 : integer);


var  diff, error : real;


X,Y : Integer;
```

```
begin

diff:=(Y2-Y1)*(X2-X1);

X:=X1; Y:=Y1;

  Putpixel(X,Y, Color); error:=0;

while X=0.5 then

begin

Y:=Y+1; Error:=Error-1;

end;

  X:=X+1; PutPixel(X,Y,Color);

end;
```

```
end;
```

Алгоритъм на Брезенхайм е усъвършенстван модел на предният алгоритъм, като тук сравнението става не с S както беше в предният алгоритъм а с X/2. Ето и алгоритъма реализиран на Pascal:

```
Procedure Brez(X1,Y1,X2,Y2 : integer);

var  error,X,Y, dX, dY : integer;

begin

d                X:=X2-X1; dY:=Y2-Y1;

Error:=-(Dx div 2); X:=X1;

Y:=Y1; Putpixel(X,Y,Color);
```

```
While X=0 then

begin

Y:=Y+1;Error:=Error-dX;

end;

X:=X+1; PutPixel(X,Y, color);

end;

end;
```

Тази процедура е само за I октан, ако искаме да е в сила за всички октани, то трябва да правим проверка къде X и Y си сменят абсолютните стойност.

ТЕМА 6 :

За генериране на коничните сечения могат да се използват техните параметрични уравнения. Системата параметрични уравнения с които се дефинира елипса има следния вид:

$x=a.\cos\varphi$

$y=b.\sin\varphi$, където φ се изменя от 0 до 2π.

Ако a стане равно на b елипсата добива вид на окръжност.

При изместване на центъра на елипсата от центъра на координатната система нейните параметрични уравнения се променят по следната зависимост:

$x=a.\cos\varphi.\cos\alpha-b.\sin\varphi.\cos\alpha+x_C$

$y= a.\cos\varphi.\sin\alpha+b.\sin\varphi.\sin\alpha+y_C$

Но ъгъла не е константа, защото се определя от правата минаваща през центъра на елипсата и пресичаща оста OX. Ако центъра на елипсата лежи на оста OX но не съвпада с началото на координатната система, тогава можем да пренебрегнем ъгъла и уравненията ще изглеждат така:

$x=A.\cos\varphi - B.\sin\varphi + x_C$

$y=C.\cos\varphi + D.\sin\varphi + y_C$

Където $A=a.\cos\varphi$; $B=b.\sin\varphi$; $C=a.\sin\varphi$; $D=b.\cos\varphi$;

Както ни е известно от математиката окръжността може да се опише като правилен N-ъгълник където N клони към безкрайност. При представяне на окръжност в компютъра, средсвата му за визуализация са недостатъчно прецизни, ето защо N може значително да се намали като достигне до 360. Параметричното уравнение на окръжност се представя по следния начин:

$x=R.\cos\varphi$, $y=R.\sin\varphi$, където R е радиуса а е в диапазона $[0..2\varphi]$. Ето една примерна процедура за реализиране на окръжност на Pascal:

```
Procedure Circle;

const thine=2+ /100;

var x1,y1,x2,y2,I:integer;

th:real;

begin

th:=0; x1:=x0+R: y1:=y0;

for I:= 1 to 100 do

begin

th:=th+thine;
```

```
x2:=x0+raond(R*cos(th));

y2:=y0+raond(R*sin(th));

line(x1,y1,x2,y2);

x1:=x2; y1:=y2;

end;

end;
```

В тази процедура не се прави нищо за апроксимиране на окръжността, и ако броя на страните на многоъгълника е малък може да не се вижда окръжност а многоъгълник. Може да се апроксимира окръжност и само с точки което е по=качествения вариант.

ТЕМА 7 :

Запълнената област се представя със затворен многоъгълник, който се запълва с произволен цвят или по някои от следните начини:

- Празна област - като не се запълва а само се изчертава контур;

- Изцяло запълване - вътрешността на контура се запълва със зададен цвят;

- Запълване с графична фигура;

- Щриховане;

Векторните графични устройства запълват образите като изчертават множество от отсечки разпростиращи се в рамките на многоъгълника. Алгоритмите се основават на пресечните точки на две отсечки, т.е. техният анализ.

Растерните графични устройства има добри възможности при изграждането на запълнена област. Разработени са много алгоритми, някои от които анализират ограждащите отсечки на многоъгълника, и определят кои пиксел са вътре и кои извън него, други започват установяването на пиксели във вътрешността и анализират околните пиксели.

Метода на сканиращата линия анализира пресичането на контурните отсечки с въображаема линия, която се премества по една от осите. Всички пиксели по нейното протежение, принадлежащи на областта се оцветяват в даден цвят. Разглеждат се и пресечните точки на всички сканиращи линии и контурните отсечки. Подредени от ляво на дясно, пресечните точки се сортират по двойки, линията м/у всяка двойка се запълва с пиксели. Когато сканиращата линия пресича връх на полигона, в списъка на пресечните точки се записва две еднакви стойности. За конкретно запълване на областта трябва да се разгледа целият полигон. При този метод областта се обработва от горе надолу, и от ляво на дясно. Описаните до тук действия важат само за изпъкнали многоъгълници. Ако имаме вдлъбнат многоъгълник то тя не би работила коректно. Ето защо се налага да следим и дали отсечката която пресичаме е монотонно намаляваща или монотонно растяща. В такъв случай запълването се осъществява от монотонно намаляваща до такава растяща отсечка, по дължината на СЛ. Така вече сме независими от типа на многоъгълника.

Друг тип метод за запълване на област, е така нареченият метод на запълване от вътре навън. Установява се един пиксел и се анализират съседните му. Ако те не принадлежат към границите

на областта се оцветяват с цвета на запълване, избира се следващ пиксел, анализира се и така докато не запълним обекта. Този метод е особено приложим в илюстративната графика и компютърното рисуване. Потребителят интерактивно скицира границите на областта с определен цвят, посочва една вътрешна точка, след което алгоритъма анализира дали околните и пиксели са с цвета на граничните отсечки. Ако не са, се установяват с цвета на запълването, който е различен от цвета на границите, и така до пълното запълване на областта. Могат да се проверяват само 4 околни пиксела, или и осемте възможни.

ТЕМА 8:

За да можем да преминем в графичен режим и да чертаем дадени обекти, то ние първо трябва да инициализираме графичната система. В Pascal това става чрез процедурата InitGraph(grDriver, grMode , Пътя до драйвера).

GrMode може да има следните стойности:

Вид монитор

Номер

Current Driver

-128

Detect

0

CGA

1

MCGE

2

EGA

3

EGA64

4

EGAMono

5

Herc mono

6

IBM 8414

7

ATT 400

8

VGA

9

PC3270

10

Ако всичко е минало безпроблемно, би трябвало стойността на ErrCode=0, в противен случай неговата стойност носи номера на грешката получена при опита за инициализация. Дефинирани са следните кодове за грешки, които може да върне ErrCode:

0

Няма грешка

1

Неинициализирана графична система

2

Не е установен графичен хардуер

3

Не е открит файл с графичен драйвер

4

Невалиден файл с графичен драйвер

5

Недостатъчна памет за драйвера

6

Недостатъчна памет при ск. запълване

7

Малко памет при нав. запълване

8

Не е открит файл с графичен шрифт

9

Недостатъчна памет за шрифта

10

Невалиден гр. режим за избр. драйвер

11

Малко памет в табл. на шрифтовете

12

Графична грешка при вход/изход

13

Невалиден файл с шрифт

14

Невалиден номер на шрифта

За изчертаване на точка се използва PutPixel(x,y,color), за изчертаване на отсечка се използва Line(x1,y1,x2,y2), Çà çàäàâàíå íà öâÿò ñå èçïîëçâà setcolor(color).

За изчертаване на многоъгълник, DrawPoly(брой на точки, масив с координати), за изчертаване на окръжност се използва Circle(x1,y1,Radius), за да препозиционираме курсора се използва MoveXY(x,y), за текст Outtext(Text). Последните процедури чертаят с текущите стойности на зададеният цвят и запълване(Pattern); Можем да променяме запълването с процедурата SetFillStyle(Pattern, color), като можем ние да си определяме типа на Pattern чрез процедурата SetFillPattern(); Ако искаме да узнаем с какви стойности на цвета и фона работим в момента то можем да използваме функциите:

GetColor, GetFillPattern, GetFillStyle, GetPixel(x,y); и т.н. В езика C++

повечето процедури и функции са идентични на тези в Pascal. Когато завършим работата си с графичната система, ние трябва да я затворим Това става с функцията CloseGraph.

ТЕМА 9:

След като имаме един графичен обект, върху него могат да се прилагат различни методи за промяна на състоянието му. Възприето координатите на даден обект да се записват в матрица, за да е по лесна работата ни с тях. Съответно върху един обект могат да се прилагат няколко вида трансформации, най често използваните от които са описани по-надолу:

1.Транслация (преместване) - За транслиране на точка с координати (X,Y) в точка с координати (X',Y'), с вектор на транслация (M,N), се използват формулите :
X' = X + M и Y' = Y + N

Както казахме, простите двумерни трансформации, могат да се представят с матрици [3x3], координатите на точката се записват в матрица с 3 колони по следният начин A (X, Y, 1). Последното се прави с цел да се изравнят матриците за по-лесно смятане. Ето я и самата транслираща матрица с вектор M,N:

1

0

0

0

1

0

M

N

1

2. Ротация около началото на координатната система , и с

ъгъл - За ротация на точка с координати (X,Y) в точка с координати (X',Y'), с ъгъл , се използват формулите :
X' = X.cos() - Y.sin() и Y' = X.sin() - Y.cos()

А матрицата на ротация има вида:

cos()

sin()

0

-sin()

cos()

0

0

0

1

3. Мащабиране с коефициенти Kx, Ky:

Kx

0

0

0

Ky

0

0

0

1

4. Матрица на симетрия относно:

-OX

1

0

0

0

-1

0

0

0

1

-OY

-1

0

0

0

1

0

0

0

1

- Спрямо права Y=X

0

1

0

1

0

0

0

0

1