ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет автоматики и вычислительной техники
Кафедра «Электронные вычислительные машины»

С.Д. Блинова

КОМПЛЕКС ЛАБОРАТОРНЫХ РАБОТ 1-13

Методические указания
по выполнению лабораторных работ

Дисциплина
«Компьютерная графика»

Специальность 09.03.01
«Информатика и вычислительная техника»

Киров,2015

Печатается по решению редакционно-издательского совета
Вятского государственного университета

УДК 004.92(07)
Б

Рецензент: доктор технических наук, профессор кафедры РЭС А.В. Частиков

Блинова С,Д, Комплекс лабораторных работ. Методические указания по выполнению лабораторных работ/ С.Д. Блинова. – Киров: Изд-во ВятГУ, 2015. – 40 с.

Редактор Е.Г. Козвонина

Подписано в печать Усл.печ.л. 1,13
Бумага офсетная Печать копир Aficio 1022
Заказ №441 Тираж 27 Бесплатно
Текст напечатан с оригинал-макета, представленного автором.

610000, г.Киров, ул. Московская, 36.
Оформление обложки, изготовление – ПРИП ВятГУ

Содержание

Лабораторные работы №1,2,3.
Реализация алгоритмов растровой развертки линий

Цель работы: закрепить лекционный материал по изучению базовых алгоритмов компьютерной графики - разложению отрезков и окружностей в растр.

Краткие теоретические сведения

Поскольку используемые дисплеи - растровые, для представления геометрического объекта на экране необходимо уметь получать его дискретное или растровое представление. Это означает, что заданной геометрической фигуре следует поставить в соответствие множество точек растра, которое в некотором смысле является приближением этой фигуры. Такое представление неоднозначно, т.к. существуют разные понятия и способы приближения непрерывного объекта. Например, растровым приближением круга на плоскости может служить множество внутренних точек круга, а прямой - множество точек - центров ячеек растра, с которыми эта прямая пересекается. Важным является эффективность того или иного представления.

Существуют стандартные процедуры программной или аппаратной реализации алгоритмов генерации линий и закраски многоугольников в различных графических пакетах, но для студентов специальности 220100 необходимо представлять и уметь программно реализовывать такие алгоритмы при работе с устройствами типа дисплея, принтера, плоттера. Необходимость использовать свою версию алгоритма генерации возникает, например, когда атрибуты инициируемого пиксела зависят от каких-либо условий (положения пиксела на прямой или внутри закрашиваемого многоугольника) или когда Вас не устраивает скорость работы "стандартного" алгоритма. Такие ситуации типичны при реализации алгоритмов загораживания (например, растровой реализации одной из версий алгоритма плавающего горизонта или метода буфера глубины).

Разложение отрезка в растр

Процесс последовательной инициализации множества пикселов экрана, составляющих отрезок, называется его растровой разверткой, а само это множество - растровым представлением отрезка. Далее будем рассматривать два таких представления: восьмисвязное и четырехсвязное. Четырехсвязная развертка отрезка предполагает различие у соседних точек на отрезке только значения одной координаты. У восьмисвязной - могут отличаться обе координаты.

Простой пошаговый алгоритм. Если увеличивать с определенным шагом координату X, а затем находить координату Y используя уравнение прямой Y=mX+b и подкрашивать пиксел с координатами (X,ROUND(Y)), то потребуется много времени (на нецелочисленные операции). Если шаг по X принять равным единице, то m=dY/dX сводится к m=dY, т.е.изменение X на единицу приведет к изменению углового коэффициента на m. Т.о., если X(I+1)=X(I)+1,то

Y(I+1)=Y(I)+m. Алгоритм корректно работает только для отрезков в первом и восьмом октандах. В остальных случаях требует модификации, что предлагается сделать студентам самостоятельно в ходе выполнения данной лабораторной работы. При модификации следует учесть, что при m>1,единичный шаг по X приведет к такому увеличению Y, при котором две соседние точки на прямой расположатся далеко друг от друга. Поэтому X и Y следует поменять, чтобы увеличивать на единицу Y, а X - на dX = dY/m = 1/m.

Описание алгоритма для работы в первом и восьмом октандах :

Дано : m - угловой коэффициент(-1<m<1);
(x1,y1) (x2,y2) - координаты концов отрезка;
col - цвет для подкрашивания точек отрезка;
Нарисовать_точку (x,y,col) - процедура для подкрашивания цветом col пиксела с координатами(x,y);
Round(x) - функция округления x до ближайшего целого.
Требуется : разложить отрезок в растр.

1.Если x1 # x2 идти к 2 иначе к 4.
2.m=(y2-y1)/(x2-x1); y=y1;
3.Для x от x1 до x2 выполнить:
начало
Нарисовать_точку(x,Round(y),col);
y=y+m;
конец
4.Если y1=y2 идти к 5 иначе к 6.
5.Нарисовать_точку(x1,y1,col). Идти к 7.
6.Вывести сообщение об ошибке (вертикаль).
7.Закончить.

Целочисленные алгоритмы Брезенхема

Основной недостаток простейшего алгоритма - использование вещественных вычислений. В 1965 году Брезенхемом был предложен целочисленный алгоритм растрового построения отрезка, первоначально предназначенный для графопостроителей. Суть алгоритма в следующем: в процессе работы одна из координат либо x, либо y (в зависимости от углового коэффициента) изменяется на единицу. Изменение другой координаты (на 0 или 1) зависит от расстояния - (e) между действительным положением отрезка и ближайшими координатами растра (e назовем управляющей переменной). Алгоритм построен так, что на каждом шаге проверяется лишь знак e и корректируется ее значение после каждого изменения очередной координаты. Значение исходной управляющей переменной:

e=2*(y2-y1)-(x2-x1),

где x1,y1,x2,y2 - координаты начальной и конечной точек отрезка. В каждом шаге при e>=0 значение y от предыдущего увеличивается на единицу, а e

уменьшается на 2*(x2-x1), в противном случае - y не меняется, а значение e увеличивается на 2*(y2-y1). В обоих случаях координата x следующего пиксела увеличивается на единицу от предыдущего значения.

Описание общего алгоритма Брезенхема:

Для восьмисвязной развертки:

Дано: (x1,y1,x2,y2)-координаты начальной и конечной точек отрезка;
col - цвет пикселов отрезка;
Нарисовать_точку(x,y,col) - процедура для подкрашивания цветом col пиксела с координатами(x,y);
sign(x) - функция, возвращающая 1,если x>=0 и -1,если x<0;
round(x)- функция округления x до ближайшего целого;
abs(x) - функция, возвращающая целое от x.
Требуется : разложить отрезок в растр.

1.Присвоить: x=x1; y=y1; dx=abs(x2-x1); dy=abs(y2-y1);
s1=sign(x2-x1); s2=sign(y2-y1);
2.Если dy>dx идти к 3, иначе к 4.
3.Поменять ролями dx и dy.Переменной l присвоить значение "истина". Идти к 5.
4.l присвоить значение "ложь". Идти к 5.
5.Присвоить e исходное значение 2*dy-dx.
6.Для i от 1 до dx выполнить:
начало
Нарисовать_точку(x,y,col); Пока e>=0 выполнить:
начало если l="истина" то x=x+s1 иначе
y=y+s2;
e=e-2*dx
конец
Если l="истина" то y=y+s2 иначе x=x+s1;
e=e+2*dy
конец
7.Нарисовать_точку(x,y,col).
8.Закончить.

Для четырехсвязной развертки:

Дано: (x1,y1,x2,y2)-координаты начальной и конечной точек отрезка;
col - цвет пикселов отрезка;
Нарисовать_точку (x,y,col) - процедура для подкрашивания цветом col пиксела с координатами(x,y);
sign(x) - функция, возвращающая 1,если x>=0 и -1,если x<0;
round(x) - функция округления x до ближайшего целого;
abs(x) - функция, возвращающая целое от x;
Требуется : разложить отрезок в растр.

1.Присвоить: x=x1; y=y1; dx=abs(x2-x1); dy=abs(y2-y1);
s1=sign(x2-x1); s2=sign(y2-y1);
2.Если dy<dx то присвоить переменной l значение "ложь" и идти к 4 иначе идти к 3.
3.Поменять ролями dx и dy. Переменной l присвоить значение "истина". Идти к 4.
4.Присвоить e исходное значение 2*dy-dx.
5.Для i от 1 до dx+dy выполнить:
начало
Нарисовать_точку(x,y,col);
Если e<0 выполнить: начало
если l="истина" то y=y+s2 иначе x=x+s1;
e=e+2*dy
конец иначе выполнить:
начало
если l="истина" то x=x+s1 иначе y=y+s2;
e=e-2*dx
конец
конец
6.Нарисовать_точку(x,y,col).
7.Закончить.

Растровая развертка окружностей

Существуют несколько простых способов преобразования окружности в растровую форму. Например, по формуле X*X+Y*Y=R*R для окружности с центром в начале координат. Чтобы изобразить четверть такой ок ружности, на каждом шаге следует поменять X от 0 до R на единицу и вычислить Y как SQRT(R*R-X*X). Остальные четверти изображают симметрично. Этот метод содержит операции умножения и извлечения корня, потому не эффективен. Кроме того, при X близких к R, в окружности появляются заметные промежутки, так как при таких X тангенс угла наклона касательной к окружности стремится к бесконечности. Процесс можно улучшить, если вычислять одну восьмую часть окружности, а остальные семь частей отображать симметрично (в предыдущем случае X менять от 0 до R/SQRT(2)). Но необходимый эффект можно получить только при работе с целыми числами (алгоритмами, подобными рассмотренным выше). Для генерации окружностей Брезенхем предложил следующий алгоритм:

Дано: rad - радиус окружности
col - цвет пикселов окружности
e - управляющая переменная
Нарисовать_точку(x,y,col) - процедура для подкрашивания цветом col пиксела с координатами(x,y)
Требуется: разложить окружность в растр.

1.Присвоить:x=0;y=rad;e=3-2*rad;
2.Пока x<y выполнить:
   Нарисовать_точку(x,y,col);
   Нарисовать_точку(y,x,col);
   Нарисовать_точку(y,-x,col);
   Нарисовать_точку(x,-y,col);
   Нарисовать_точку(-x,-y,col);
   Нарисовать_точку(-y,-x,col);
   Нарисовать_точку(-y,x,col);
   Нарисовать_точку(-x,y,col)
3.Если e<0 то e=e+4*x+6 и идти к 5 иначе идти к 4.
4.Выполнить: e=e+4*(x-y)+10; y=y-1
5.x=x+1 идти к 2 пока x<y иначе к 6.
6.Если x=y то идти к 7 иначе к 8.
7.Выполнить:
   Нарисовать_точку(x,y,col);
   Нарисовать_точку(y,x,col);
   Нарисовать_точку(y,-x,col);
   Нарисовать_точку(x,-y,col);
   Нарисовать_точку(-x,-y,col);
   Нарисовать_точку(-y,-x,col);
   Нарисовать_точку(-y,x,col);
   Нарисовать_точку(-x,y,col).
8.Закончить.

Задание на лабораторную работу

1. Написать на языке PASCAL программу, реализующую алгоритмы построения прямой: простой пошаговый алгоритм и алгоритмы Брезенхема для четырех- и восьмисвязной развертки.
2. Проверить правильность работы программы, нарисовав, например, каждым алгоритмом семейство радиальных прямых, выходящих из одной точки с шагом 15 градусов.
3. Написать и отладить программу, реализующую два алгоритма построения окружности: по формуле Y=+-SQRT(r*r-x*x) и Брезенхема. В обоих случаях использовать свойство симметрии окружности (в первом - найдя точки четверти окружности, остальные - отразив симметрично; во втором - свойство симметрии использовать полностью).
4. Реализовать демонстрационный ролик (анимацию) движения объектов прямолинейно и по окружности.

Дополнительное задание

Используя программы работы с мышью и программы, написанные при выполнении пунктов 1-3 задания, вывести на экран простой рисунок из линий и окружностей.

Пример программы работы с мышью:

{

Пример использования мыши в программах на языке Паскаль. Работу с мышью обеспечивает драйвер, отслеживающий перемещения ее курсора, нажатие и отпускание кнопок. Реализация работы идет через механизм прерываний. Номер прерывания - 33h. В регистры помещаются входные и выходные параметры. В языке Паскаль для этих целей используют процедуру Intr(IntNo:Byte;varRegs:Registers) модуля Dos. Нельзя выводить изображение поверх курсора мыши. В случае необходимости нужно убрать курсор, вывести изображение, затем вывести курсор.

}

```
program mouse;
uses Crt,Graph,Dos;
var
  grDriver: Integer;
  grMode: Integer;
  ErrCode: Integer;
  X,Y,X1,Y1:Integer;
  L,M,R:Boolean;
  BL:Boolean;
{
 Инициализация и проверка наличия мыши. Возвращает истину (есть) или ложь
(нет) }
    function ReadMouse : Boolean;
   var
   r: Registers;
   begin
   r.ax:=0;
   intr($33,r);
   ReadMouse:=r.ax=$ffff;
   end;

{Визуализировать курсор мыши}
   procedure ShowMouseCursor;
   var
   r:registers;
   begin
   r.ax:=1;
   intr($33,r);
   end;

{Сделать курсор мыши невидимым (перемещение отслеживается)}
   procedure HideMouseCursor;
   var
   r:registers;
```

```
   begin
   r.ax:=2;
   intr($33,r);
   end;

  {Возврат координат положения курсора и нажатой клавиши мыши}
  procedure ReadMouseStatus(var x,y:integer;var LB,MB,RB :boolean);
   var
   r:Registers;
   begin
   r.ax:=3;
   intr($33,r);
   x:=r.cx;
   y:=r.dx;
   LB:=(r.bx And 1) <> 0;
   RB:=(r.bx And 2) <> 0;
   MB:=(r.bx And 4) <> 0;
   end;

{Перемещение курсора мыши к указанным координатам}
   procedure MoveMouseCursor(x,y:integer);
   var r:registers;
   begin
   r.ax:=4;
   r.cx:=x;
   r.dx:=y;
   intr($33,r);
   end;
{
 Основная программа. Рисует линию по двум точкам, задаваемым щелчком левой
кнопки мыши. По щелчку правой кнопки мыши - выходит из программы
}
   begin
  grDriver := detect;
  InitGraph(grDriver, grMode,'d:\tp7\bgi');
  ErrCode := GraphResult;
  if ErrCode = grOk then
                begin  { Do graphics }
  L:=false;M:=false;R:=false;Bl:=true;
SetColor(Cyan);
SetBKColor(LightGray);
if ReadMouse then begin
ShowMouseCursor;
While true do
     begin
```

```
Repeat ReadMouseStatus (X,Y,L,M,R) Until L or R;
If R then begin
CloseGraph;
Exit      end;
While L do ReadMouseStatus(X,Y,L,M,R);
if BL then
 begin
X1:=X;Y1:=Y;
Bl:=false
 end
else
  begin
HideMouseCursor;
Line(X1,Y1,X,Y);
ShowMouseCursor;
Bl:=true
  end;
      end;
                 end
 else

     begin
RestoreCrtMode;Writeln('error');Halt;
     end;
                  end
  else
    Writeln('Graphics error:', GraphErrorMsg(ErrCode));
end.
```

Требования к защите лабораторной работы

Защита лабораторной работы состоит из демонстрации преподавателю результатов выполнения заданий на лабораторную работу и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист, оформленный согласно требованиям кафедры, и листинг программы - результат выполнения заданий лабораторной работы с обязательными комментариями.

Вопросы для самоподготовки

1. Каков визуальный эффект применения простого пошагового алгоритма к отрезкам, угловой коэффициент которых больше единицы или меньше минус единицы?
2. Как избавиться от этого эффекта?
3. Почему простой пошаговый алгоритм не используют для генерации отрезков в современных графических пакетах?
4. Каков визуальный эффект при нахождении координаты Y всех точек или одной четверти окружности по формуле X^2+Y^2=R^2? Как избавиться от этого эффекта?
5. Каким образом в алгоритмах Брезенхема удается избежать нежелательных эффектов при построении точек линий?

Литература

1. Фоли, Дж. Основы интерактивной машинной графики [текст]: В 2 кн. кн. 2 / Дж. Фоли, А. Дэм; под ред. Ю. М. Баяковского; пер. с англ. В. А. Галактионова и др. - М.: Мир, 1985. - 368с.: ил.
2. Роджерс, Д. Ф. Алгоритмические основы машинной графики [текст] / Д. Ф. Роджерс; пер. с англ. С. А. Вичеса и др.; Под ред. Ю. М. Баяковского, В. А. Галактионова. - М.: Мир, 1989. - 503с.
3. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] / Е. В. Шикин, А. В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.
4. Боресков, А. В. Компьютерная графика: первое знакомство [текст] / А. В. Боресков, Е. В. Шикин, Г. Е. Шикина; Под ред. Е. В. Шикина. - М.: Финансы и статистика, 1996. - 176с.: ил. - (Диалог с компьютером).

Лабораторная работа №4,5 Реализация вывода сплайнов Безье

Цель работы: Закрепить лекционный материал по изучению материала темы "Кривые Безье". Реализовать геометрический алгоритм вывода кривых Безье.

Теоретические сведения

Историю сплайнов принято отсчитывать от момента появления первой работы Шенберга в 1946 году. Сначала сплайны рассматривались как удобный инструмент в теории и практике приближения функций. Однако довольно скоро область их применения начала быстро расширяться и обнаружилось, что существует очень много сплайнов самых разных типов. Сплайны стали активно использоваться в численных методах, в системах автоматического проектирования и автоматизации научных исследований, во многих других областях человеческой деятельности и, конечно, в компьютерной графике.

Сам термин "сплайн" происходит от английского «spline». Именно так называется гибкая полоска стали, при помощи которой чертежники проводили через заданные точки плавные кривые. В былые времена подобный способ построения плавных обводов различного рода тел, например, таких, как корпус корабля, фюзеляж или крыло самолета, кузов автомобиля и т. п., был довольно широко распространен в практике машиностроения. В результате форма тела задавалась при помощи набора очень точно изготовленных сечений – плазов.

Появление компьютеров позволило перейти от этого, плазово-шаблонного, метода к более эффективному способу задания поверхности обтекаемого тела. В основе этого подхода к описанию поверхностей лежит использование сравнительно несложных формул, позволяющих восстанавливать облик изделия с необходимой точностью. Ясно, что для большинства тел, встречающихся на практике, вряд ли возможно отыскание простых универсальных формул, которые описывали бы соответствующую поверхность глобально, т. е. в целом. Это

означает, что при решении задачи построения достаточно произвольной поверхности небольшим количеством формул, как правило, обойтись не удается. Вместе с тем аналитическое описание (описание посредством формул) внешних обводов изделия, т. е. задание двухмерной поверхности в трехмерном пространстве) должно быть достаточно экономным. Это особенно важно, когда речь идет об обработке изделия на станках с числовым программным управлением. Обычно поступают следующим образом: задают координаты сравнительно небольшого числа опорных точек, лежащих на искомой поверхности, и через эти точки проводят плавные поверхности. Именно так поступает конструктор при проектировании кузова автомобиля. На следующем шаге конструктор должен получить аналитическое представление для придуманных кривых или поверхностей. Вот для таких задач и используются сплайны.

Средства компьютерной графики, особенно визуализация, существенно помогают при проектировании, показывая конструктору, что может получаться в результате, и, давая ему многовариантную возможность сравнить это с тем, что есть у него в голове.

Сплайн Безье [2]

Разработан математиком Пьером Безье. Кривые (сплайны) и поверхности Безье были использованы в 60-х годах компанией "Рено" для компьютерного проектирования формы кузовов автомобилей. В настоящее время они широко используются в компьютерной графике. Кривые Безье описываются в параметрической форме:

$x=P_x(t)$,

$y=P_y(t)$.

Значение t выступает как параметр, которому отвечают координаты отдельной точки линии. Параметрическая форма описания может быть более

удобной для некоторых кривых, чем задание в виде функции y=f(x). Это потому, что функция f(x) может быть намного сложнее, чем $P_x(t)$ и $P_y(t)$, кроме того, f(x) может быть неоднозначной.

Многочлены Безье для Px(t) и Py(t) имеют такой вид:

$$Px(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} x_i$$

$$Py(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i$$

где C_m^i - сочетание m по i (известное также по биному Ньютона),

$$C_m^i = \frac{m!}{i! * (m-i)!}$$

а x_i и y_i координаты точек-ориентиров P_i.

Значение m можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров.

Геометрический алгоритм для кривой Безье [2]

1. Каждая сторона контура многоугольника, проходящего по точкам-ориентирам, делится пропорционально значению t.
2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предыдущего контура.
3. Стороны нового контура снова делятся пропорционально значению t.

И так далее. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье. Пример построения кривой Безье, используя геометрический алгоритм, приведён на рисунке 1.

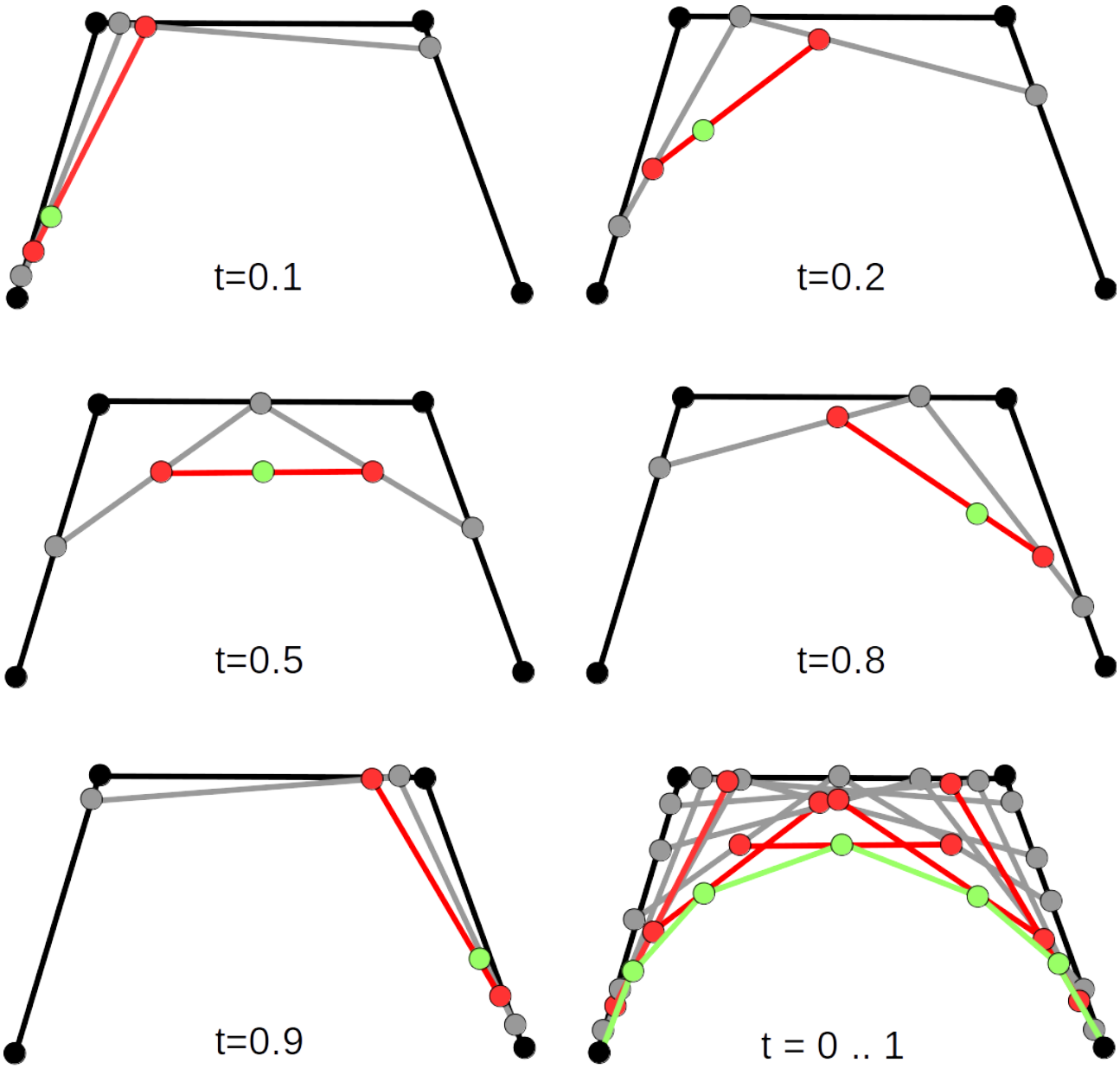

Рисунок 1 – Результата работы геометрического алгоритма построения кривой Безье

Реализация геометрического алгоритма

Исходные данные:

step – шаг увеличения параметра t, определяет количество точек, по которым будет построена кривая Безъе;

m – количество точек исходного многоугольника;

R - массив с координатами точек нового многоугольника;

P - массив с координатами точек исходного многоугольника;

x_n, y_n – координаты первой точки линии, выводящейся на каждом шаге работы программы.

1. x_n := P[1].x; y_n := P[1].y; t := 0; step := 0.01;
2. Присвоить новому многоугольнику координаты исходного: R:= P;
3. Для j от m до 2 делать: для i от 1 до (j-1) делать:

Выполнить поиск координат точек нового многоугольника по формулам:

R[i].x := R[i].x + round(t * (R[i + 1].x-R[i].x));

R[i].y := R[i].y + round(t*(R[i + 1].y-R[i].y));

4. Нарисовать отрезок с координатами (xn, yn, R[1].x, R[1].y);
6. t := t + step; x_n := R[1].x; y_n := R[1].y;
7. Если t > 1, то идти к пункту 7, иначе к пункту 2;
8. Конец.

Задание на лабораторную работу

1. Написать на языке PASCAL программу, реализующую геометрический алгоритм построения кривой Безъе. Кривая должна строиться пошагово (с задержкой), отображая вспомогательные многоугольники, используемые для получения каждой точки.

2. Реализовать демонстрацию анимации движения объектов по сложным траекториям, составленным из кривых Безье.

Список литературы

1. Шишкин Е.В., Боресков А.В., Зайцев А.А. Начала компьютерной графики, // «Диалог-МИФИ» Москва 1993;

2. Порев В.Н. Компьютерная графика. - Спб.: БХВ-Петербург, 2002 - 432 с.: ил..

Лабораторная работа № 6,7
Аффинные преобразования на плоскости

Цель работы: Закрепить лекционный материал по изучению материала одноименной темы, реализовав матрицы переноса, масштабирования, отражения и вращения применительно к координатам описанной в программе плоской фигуры (многоугольника) с целью демонстрации движения и преобразования формы этой фигуры на плоскости.

Краткие теоретические сведения

Система координат - совокупность правил, ставящих в соответствие каждой точке набор чисел(координат), число которых называется размерностью пространства. Допустим, на плоскости введена прямолинейная координатная система, тогда каждой точке ставится в соответствие упорядоченная пара чисел (x,y) - ее координат.

В компьютерной графике чаще всего используют аффинную и декартовую систему координат. Это прямоугольная координатная система, в которой выбирается точка O - начало координат и два приложенных к ней неколлинеарных единичных вектора ex и ey, которые задают оси координат. Если единичные отрезки на осях не равны, система называется аффинной, если равны и угол между осями координат прямой - прямоугольной декартовой.

Однородным представлением n-мерного объекта является его представление в (n+1)-мерном пространстве, полученное добавлением еще одной координаты - скалярного множителя. При решении задач компьютерной графики однородные координаты обычно вводятся так: произвольной точке M(x,y) на плоскости ставится в соответствие точка M(x,y,1) в пространстве.

Если нам необходимо преобразовать точку на плоскости с координатами (x,y) в другую точку то задача сводится к поиску новых координат для этой точки - (x). В случае аффинных преобразований такой поиск сведется к решению уравнений

$$x' = ax + by + m$$
$$y' = cx + dy + n,$$

где a, b, c, d, m, n - произвольные числа, причем: $\begin{bmatrix} ab \\ cd \end{bmatrix} \approx 0$.

В случае, когда m и n не равны нулю, для представления преобразования в матричной форме нужно исходные и преобразованные координаты точки записать в однородных координатах (x,y,1) и ($x', y', 1$). Тогда в матричной форме общий вид преобразования будет следующим

$$[x'\ y\ 1] = [x\ y\ 1] * \begin{bmatrix} a\ c\ 0 \\ b\ d\ 0 \\ m\ n\ 1 \end{bmatrix} = [a*x + b*y + m \quad c*x + d*y + n \quad 1]$$

Наибольшее распространение получили частные случаи аффинных преобразований:

1. Единичное преобразование. Единичная матрица оставляет точку неподвижной. $[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad 1]$

2. Сдвиг или плоско-параллельный перенос. Матрица переводит точку на m единиц вдоль оси x и на n -вдоль оси y: $[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix} = [x+m \quad y+n \quad 1]$.

3. Вращение вокруг начала координат. Его матрица осуществляет поворот точки объекта на угол g против часовой стрелки: $[x'\ y'\ 1] = [x\ y\ 1] * \begin{bmatrix} \cos g & \sin g & 0 \\ -\sin g & \cos g & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x*\cos g - y*\sin g \quad x*\sin g + y*\cos g \quad 1]$.

4. Вращение вокруг произвольного центра осуществляет поворот вокруг точки (m,n) на угол g против часовой стрелки. Преобразование выполняется как последовательность трех элементарных:

- сдвиг центра вращения (m,n) в начало координат с помощью матрицы сдвига
- поворот на угол g вокруг начала координат с помощью матрицы вращения
- сдвиг точки (m,n) в исходное положение, используя матрицу сдвига

Итоговое преобразование будет выглядеть так:

$$[x'\ y'\ 1] = [x\ y\ 1] * \begin{bmatrix} \cos g & \sin g & 0 \\ -\sin g & \cos g & 0 \\ -m*\cos g + m + n*\sin g & -m*\sin g - n*\cos g + n & 1 \end{bmatrix} =$$

$$[x*\cos g - y*\sin g - m*\cos g + m + n*\sin g \quad x*\sin g + y*\cos g - m*\sin g - n*\cos g + n \quad 1]$$

5.Симметрия относительно оси, проходящей через начало координат осуществляется преобразованием вида

$$[x'\ y'\ 1] = [x\ y\ 1] * \begin{bmatrix} \cos g & \sin g & 0 \\ \sin g & -\cos g & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x*\cos g + y*\sin g \quad x*\sin g - y*\cos y \quad 1]$$

При этом, если угол между осью симметрии и осью Ox = w, то угол g=2*w. Согласно этому, симметрия относительно оси Ox

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x \quad -y \quad 1],$$

оси Оу $$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-x \quad y \quad 1],$$

оси y=x $$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [y \quad x \quad 1]$$

оси y=-x $$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-y \quad -x \quad 1].$$

6.Масштабирование – увеличение (уменьшение размеров изображения) - в общем случае изменяет форму объекта. Назначается точка, относительно которой производится преобразование.

Масштабирование относительно точки О - начала координат

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} kx & 0 & 0 \\ 0 & ky & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x * kx \quad y * ky \quad 1],$$

где kx ,ky - коэффициенты искажения по осям Ox, Oy соответственно.

При kx=ky=k осуществляется преобразование подобия, при kx # ky изображение искажается. Изображение увеличивается при k > 1 и уменьшается при k < 1.

Масштабирование относительно произвольной точки с координатами (m, n)

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] * \begin{bmatrix} kx & 0 & 0 \\ 0 & ky & 0 \\ -m * kx + m & -n * ky + n & 1 \end{bmatrix} = [x * kx - m * kx + m \quad y * ky - n * ky + n \quad 1].$$

Матрица любого аффинного преобразования может быть получена умножением соответствующих рассмотренных здесь простых матриц. Порядок умножения имеет значение, поэтому выполнять его надо в определенной логической последовательности.

Если координаты точек объекта представлены вектором - столбцом, а не вектором - строкой, соответствующая матрица для преобразований должна быть транспонирована.

Для возвращения к исходному состоянию следует использовать матрицу обратную той, что использовалась при преобразовании.

Для эффективной работы с преобразованиями следует руководствоваться следующими рекомендациями :

- лучше умножение на результирующую матрицу, чем последовательность умножений
- лучше масштабировать, а потом поворачивать, чем поворачивать, а затем масштабировать
- если комбинированное преобразование содержит поворот, то его следует делать отдельно и последним
- для того, чтобы движение казалось непрерывным и плавным, следует выводить кадры достаточно быстро (30-60 мсек) и координаты каждой точки преобразовывать быстрее.

В уравнении для поворота, например, если угол поворота g равен нескольким градусам, то cos g можно принять равным единице, тогда $\begin{matrix} x' = x - y * \sin g \\ y' = x * \sin g + y. \end{matrix}$ Еще лучше $\begin{matrix} x' = x - y * \sin g \\ y' = x' * \sin g + y. \end{matrix}$

Для того, чтобы не накапливалась ошибка округлений, следует сохранять и после каждого поворота на 360 градусов использовать исходные координаты преобразуемых точек.

Задание на лабораторную работу:

Написать на языке PASCAL программу:

1. Рисующую многоугольник.
2. Смещающую его на n пикселов вправо и m вниз.
3. Зеркально отражающую его относительно осей координат.
4. Растягивающую (сжимающую) его вдоль координатных осей относительно некоторой заданной точки.
5. Вращающую его относительно центра с координатами (k,l) по часовой стрелке с увеличением размеров, против - с уменьшением.
6. Реализовать интерактивную анимацию (взаимодействующую с пользователем), на которой выполняется поворот и масштабирование объектов.

Дополнительно

Задание выполнить с помощью программ работы с мышью.

Требования к защите лабораторной работы

Защита лабораторной работы состоит из демонстрации преподавателю результатов выполнения задания на лабораторную работу и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист с названием работы, оформленный согласно требованиям кафедры, распечатку программы - результат выполнения заданий лабораторной работы с обязательными комментариями.

Вопросы для самоподготовки:

1. Что такое аффинные преобразования и однородные координаты? Где и для чего однородные координаты используются в компьютерной графике?
2. Можно ли и каким образом, при помощи троек однородных координат и матриц третьего порядка описать любое аффинное преобразование плоскости?
3. Какие системы координат используются в компьютерной графике? Чем различаются мировая и приборная системы координат? Что такое абсолютные и относительные координаты, когда применяются те и другие?
4. Перечислите основные свойства аффинных преобразований, снискавшие им широкое распространение,
5. Чем характерна декартова система координат?

Литература

1. Фоли, Дж. Основы интерактивной машинной графики [текст]: В 2 кн. кн. 2 / Дж. Фоли, А. Дэм; под ред. Ю. М. Баяковского; пер. с англ. В. А. Галактионова и др. - М.: Мир, 1985. - 368с.: ил.
2. Роджерс, Д. Ф. Алгоритмические основы машинной графики [текст] / Д. Ф. Роджерс; пер. с англ. С. А. Вичеса и др.; Под ред. Ю. М. Баяковского, В. А. Галактионова. - М.: Мир, 1989. - 503с.
3. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] / Е. В. Шикин, А. В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.
4. Боресков, А. В. Компьютерная графика: первое знакомство [текст] / А. В. Боресков, Е. В. Шикин, Г. Е. Шикина; Под ред. Е. В. Шикина. - М.: Финансы и статистика, 1996. - 176с.: ил. - (Диалог с компьютером).

Лабораторные работы № 8,9
Реализация алгоритмов растровой графики для заполнения сплошных областей

Цель работы: Закрепить лекционный материал по изучению базовых алгоритмов компьютерной графики алгоритмов закраски.

Краткие теоретические сведения

Возможность представления сплошных областей - это одна из уникальных характеристик растрового дисплея. Все возможные методы такого представления можно разделить на две категории: растровая развертка и затравочное заполнение.

В первой в порядке сканирования строк определяют, лежит ли точка внутри области. Такие алгоритмы применимы и к векторным дисплеям для штриховки контуров.

Во второй предполагают известной некоторую точку внутри замкнутого контура. Ее называют затравочной и ищут соседние с ней точки. Если найденная точка не внутри контура, значит она принадлежит границе, если внутри - она становится новой затравочной и поиск продолжается. Подобные алгоритмы применимы только к растровым дисплеям.

Многие замкнутые контуры либо многоугольники, либо их можно аппроксимировать простыми многоугольниками. Поэтому далее речь идет о заполнении многоугольников.

Растровая развертка многоугольников

Под многоугольником будем понимать фигуру на плоскости, ограниченную несамопересекающейся замкнутой ломаной. Основная идея растровой развертки многоугольников заключается в заполнении лишь одной строки в каждый данный момент. Алгоритмы, работающие по этому принципу называют алгоритмами построчного сканирования .Простым способом выделения пикселов сканирующей строки попадающих внутрь многоугольника, является поочередная проверка всех пикселов строки растра (экрана).Но обычно большая часть пикселов лежит вне многоугольника, поэтому данный метод неэффективен во времени. Его можно несколько улучшить, помещая многоугольник внутрь минимально объемлющего прямоугольника со сторонами, параллельными сторонам растра,и анализируя лишь точки внутри этого прямоугольника .Но можно найти более эффективный метод, если допустить с большой долей вероятности, что соседние пикселы в строке (столбце) имеют одинаковые характеристики. Это допущение основывается на понятии пространственной когерентности: при перемещении от точки к точке или от одной сканирующей строки к соседней рассматриваемый многоугольник

в большинстве случаев остается постоянным. Можно искать только те точки, где когерентность строк (столбцов) нарушена.

Пусть многоугольник задан набором своих вершин, причем соседние точки в наборе - его смежные вершины. Тогда на каждой сканирующей строке внутренние пикселы можно закрасить так:

1. Найти точки пересечения сканирующей строки со всеми ребрами многоугольника, кроме горизонтальных.
2. Отсортировать найденные точки по возрастанию координаты x.
3. Закрасить все пикселы между парами точек пересечений. Алгоритм работает только в случае существенных пересечений и четного количества точек пересечений. Нечетное количество получается при пересечении сканирующей строки с локальными минимумами и максимумами, когда Y - координаты предыдущей и последующей вершин больше (меньше) чем у рассматриваемой. В этом случае такие пересечения учитываются как два. После такой корректировки алгоритм будет работать правильно при любой форме многоугольника.

Для выпуклых многоугольников алгоритм можно упростить и повысить его эффективность. Границу выпуклого многоугольника разбивают на две ломаные: "левую" и "правую" и, возможно, два ребра "верхнее" и "нижнее", тогда каждая ломаная будет иметь ровно одно пересечение с каждой сканирующей прямой. Используя алгоритм Брезенхема и одновременно генерируя растровое представление обеих ломаных, находят левый и правый пикселы границы многоугольника на каждой сканирующей прямой. Последовательно заполняя интервалы между этими пикселами от верхней строки к нижней, получают итоговую растровую развертку выпуклого многоугольника.

Алгоритмы заполнения с затравкой

Ищут и закрашивают связную компоненту области, содержащую затравочный пиксел. Под связностью понимают четырех или восьмисвязность в зависимости от постановки задачи. Каждый из пикселов четырехсвязной области достижим из любого другого пиксела этой области с помощью последовательности перемещений на пиксел из четырехэлементного набора горизонтальных и вертикальных направлений, восьмисвязной - из восьмиэлементного (добавляются еще и диагональные направления). Алгоритмы для восьмисвязных областей применимы к четырехсвязным, но не наоборот.

Процесс заполнения областей иногда называют заливкой, так как можно представить, что в точке затравки находится источник, заливающий всю область определенным цветом. Область может быть определена внутренне (все внутренние пикселы - одного цвета, внешние - другого) и гранично (все пикселы на границе или вне области - одного цвета не такого, как внутри).Далее будет рассмотрен гранично-заполняющий алгоритм для четырехсвязных областей (соответствующие внутренне-заполняющие алгоритмы можно получить аналогичным путем, а алгоритмы для четырехсвязной области легко переделать в алгоритмы для восьмисвязной).

Построчный алгоритм заливки

Применяется идея построчного сканирования и используется стек для хранения одного затравочного пиксела для любого непрерывного интервала на сканирующей строке. Непрерывный интервал - это группа примыкающих друг к другу пикселов, ограниченная уже закрашенными или граничными пикселами. Если непрерывный интервал пикселов принадлежит внутренней части области, пикселы на и под этим интервалом либо граничные, либо тоже находятся внутри области и могут служить затравочными для своих строк. Поэтому можно предложить следующую схему заполнения:

Описание алгоритма:

Дано : Pop(x,y) -процедура извлечения из стека координат (x,y) очередного пиксела
Push(x,y) - процедура помещения в стек координат (x,y) очередного пиксела
Нарисовать_точку(x,y,col) - процедура для подкрашивания цветом col пиксела с координатами(x,y)
c(x,y) - цвет пиксела с координатами (x,y)
cb - цвет для подкрашивания пикселов границы области
ci - цвет для подкрашивания пикселов внутри области
(x0,y0) - координаты затравочного пиксела
Получить : Перекрасить сплошную область с данным затравочным пикселом и цветом границы в цвет ci.

1.Инициализировать стек:Push(x0,y0).
2.Пока стек не пуст выполнить: начало
Извлечь пиксел из стека и закрасить его в цвет области:
Pop(x,y);
Нарисовать_точку(x,y,ci);
3.Присвоить: xw=x; x=x+1.
4.Пока c(x,y) # cb заполняем интервал справа от затравки:
Нарисовать_точку(x,y,ci); x=x+1;по окончанию идти к 5.
5.Присвоить: xr=x-1; x=xw; x=x-1.
6.Пока c(x,y) # cb заполнть интервал слева от затравки:
Нарисовать_точку(x,y,ci);x=x-1; по окончанию идти к 7.
7.Сохранить крайний слева пиксел: xl=x+1;
8.Проверить строки ниже и выше данной, если там есть еще незаполненные пикселы искать затравку, начиная с левого края:

Для j от -1 до 2 с шагом 3 выполнить: начало
x=xl; y=y+j;
Пока x <= xr искать затравку на строке ниже(выше): начало

fl="ложь";
Пока (c(x,y)#cb and c(x,y)#ci and x<xr) заполнить точки внутри:
начало
Увеличить x на единицу.
Если (not fl) = "истина" то присвоить fl="истина". конец
Если fl="истина" то крайний справа пиксел - в стек: начало
Если (x=xr and c(x,y)#cb and c(x,y)#ci) то Push(x,y) иначе Push(x-1,y).
Присвоить fl="ложь". конец
Продолжить проверку,если интервал был прерван: Присвоить xb=x.
Пока (c(x,y)=cb or c(x,y)=ci and x<xr) увеличить x на 1. Если x=xb то
присвоить: x=x+1.
конец
конец
конец пункта 2.
6.Закончить.

Алгоритм правильно заполняет любую область, справляется с внутренними дырами и зубцами на границе. Используется при закраске, когда известна хотя бы одна точка внутри необходимой для закрашивания области.

Задание на лабораторную работу:

1. Написать на языке PASCAL программу, реализующую алгоритм заливки многоугольника любой формы.
2. Проверить правильность работы программы, нарисовав с помощью функций модуля GRAPH невыпуклый многоугольник (с дырами внутри), закрасьте его заданным цветом, указав координаты принадлежащей многоугольнику точки.
3. Написать и отладить программу, реализующую алгоритм построчного заполнения выпуклого многоугольника, заданного координатами вершин и цветом границы (можно воспользоваться алгоритмом заполнения треугольника, прочитанным на лекции).

Дополнительное задание

Пункты 2,3 задания выполнить с помощью программ работы с мышью.

Требования к защите лабораторной работы

Защита лабораторной работы состоит из демонстрации преподавателю результатов выполнения заданий на лабораторной работы и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист с названием работы, оформленный согласно требованиям кафедры,

распечатку программы - результат выполнения заданий лабораторной работы с обязательными комментариями.

Вопросы для самоподготовки

1. Каким образом можно заполнить многоугольник?
2. Какие проблемы возникают при заполнении невыпуклого многоугольника, когда неизвестна ни одна точка, ему принадлежащая?
3. Как при заполнении сплошных областей можно использовать стек?
4. Что такое когерентность по строкам и столбцам и как, используя это свойство закрашиваемых областей, можно улучшить алгоритм закраски, сделав его более эффективным?
5. Чем отличаются алгоритмы построчного сканирования от алгоритмов заливки?

Литература

1. Фоли, Дж. Основы интерактивной машинной графики [текст]: В 2 кн. кн. 2 / Дж. Фоли, А. Дэм; под ред. Ю. М. Баяковского; пер. с англ. В. А. Галактионова и др. - М.: Мир, 1985. - 368с.: ил.
2. Роджерс, Д. Ф. Алгоритмические основы машинной графики [текст] / Д. Ф. Роджерс; пер. с англ. С. А. Вичеса и др.; Под ред. Ю. М. Баяковского, В. А. Галактионова. - М.: Мир, 1989. - 503с.
5. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] / Е. В. Шикин, А. В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.
6. Боресков, А. В. Компьютерная графика: первое знакомство [текст] / А. В. Боресков, Е. В. Шикин, Г. Е. Шикина; Под ред. Е. В. Шикина. - М.: Финансы и статистика, 1996. - 176с.: ил. - (Диалог с компьютером).

Лабораторная работа № 10
Реализация двумерного алгоритма отсечения отрезка Сазерленда-Коэна

Цель работы: Закрепить лекционный материал по изучению материала темы "Отсечение". Реализовать двумерный алгоритм Сазерленда - Коэна для отсечения отрезка окном прямоугольной формы.

Краткие теоретические сведения

Необходимость отсечь выводимое изображение по границам некоторой области (окна) встречается довольно часто, особенно в многооконных диалоговых графических системах. В простейших случаях в качестве отсекающей области выступает прямоугольник. Существует много алгоритмов отсечения для 2D- и 3D-случаев, ориентированных как на программную, так и аппаратную реализацию. Рассмотрим простой и эффективный алгоритм отсечения отрезков границами произвольного прямоугольника. Вся плоскость вывода разбивается прямыми, образующими прямоугольник на девять подплоскостей; каждой из них присваивается четырехбитовый код, в котором единица

в нулевом бите означает, что конец отрезка лежит левее окна,

в первом бите - выше окна,

во втором бите - правее окна,

в третьем бите - ниже окна.

Например, если один конец отсекаемого отрезка лежит в правом верхнем относительно окна углу экрана, он получает код этого угла равный 0110 по схеме:

0011	0010	0110
0001	0000	0100
1000	1000	1100

Описание алгоритма Сазерленда - Коэна:

Дано: прямоугольное окно с координатами (xl,ya),(xr,yb) верхнего левого и нижнего правого углов прямоугольника; отрезок с координатами концов (x1,y1) и (x2,y2).

Найти: новые координаты отрезка (x1,y1) и нарисовать его целиком или частично в случае попадания в область окна и не рисовать в противном случае.

1. Присвоить коды (p1 и p2) концам отрезка.
2. Если отрезок полностью невидим(p1 and p2 <> 0), идти к 9, иначе идти к 3.
3. Если отрезок видим полностью (p1 = 0 and p2 = 0) - нарисовать его (провести линию от точки с координатами (x1,y1) до точки - (x2,y2) и идти к 9, иначе идти к 4.
4. Если p1=0 - поменять местами точки (x1,y1) и (x2,y2),чтобы обработка шла каждый раз с точки, находящейся вне окна:
x=x1,x1=x2,x2=x,y=y1,y1=y2,y2=y,p1=p2 и идти к 5, иначе идти к 5.
5. Если точка (x1,y1) - слева от окна - найти ее новые координаты по формулам: y1=y1+(y2-y1)*(xl-x1)/(x2-x1), x1=xl и идти к 1, иначе идти к 6.

6. Если точка (x1,y1) - выше окна - найти ее новые координаты по формулам: x1=x1+(x2-x1)*(ya-y1)/(y2-y1), y1=ya и идти к 1, иначе идти к 7.
7. Если точка (x1,y1)- справа от окна - найти ее новые координаты по формулам: y1=y1+(y2-y1)*(xr-x1)/(x2-x1), x1=xr и идти к 1, иначе идти к 8.
8. Если точка (x1,y1) - ниже окна - найти ее новые координаты по формулам: x1=x1+(x2-x1)*(yb-y1)/(y2-y1),y1=yb и идти к 1, иначе идти к 1.
9. Закончить.

Присваивание кода точке с координатами (x,y) (пункт 1 данного алгоритма) можно выполнить, например, следующей программой:

```
function kod(x,y,xl,ya,xr,yb:integer):byte;
var
kp:byte;
begin
kp:=0;
if x<xl kp:=kp or $01;
if y<ya kp:=kp or $02;
if x>xr kp:=kp or $04;
if y>yb kp:=kp or $08;
kod:=kp
end;
```

Здесь (xl,ya),(xr,yb) - координаты левого верхнего и правого нижнего углов окна, kod - результирующий код.

Задание на лабораторную работу

Написать на языке PASCAL программу, реализующую алгоритм Сазерленда-Коэна, отсекающий отрезок по границам прямоугольного окна. Для показа результатов работы программы нарисовать на экране окно прямоугольной формы. Задав координаты окна и отрезка, продемонстрировать отсечение отрезка по границам окна. Рассмотреть все возможные случаи расположения отрезка относительно окна.

Дополнительно

Для демонстрации работы алгоритма использовать мышь.

Требования к защите лабораторной работы

Защита лабораторной работы состоит из демонстрации преподавателю результатов выполнения задания на лабораторную работу и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист с названием работы, оформленный согласно требованиям кафедры, распечатку программы результат выполнения заданий лабораторной работы с обязательными комментариями.

Вопросы для самоподготовки:

1. Что такое отсечение и для чего оно применяется?
2. Какие алгоритмы отсечения Вы знаете?
3. Что является границей отсечения для 2D- 3D- случаев?
4. Можно ли реализовать отсечение аппаратно, если можно, то в каких случаях это делают?

Литература

7. Роджерс, Д. Ф. Алгоритмические основы машинной графики [текст] / Д. Ф. Роджерс; пер. с англ. С. А. Вичеса и др.; Под ред. Ю. М. Баяковского, В. А. Галактионова. - М.: Мир, 1989. - 503с.
8. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] / Е. В. Шикин, А. В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.

Лабораторная работа № 11.
Представление и проецирование трехмерных объектов

Цель работы: Научиться применять математический аппарат проекций для визуализации объемных геометрических тел.

Краткие теоретические сведения

Изображение объектов на экране связано с такой геометрической операцией, как проецирование. В компьютерной графике используют в основном параллельное и центральное проецирование прямыми лучами на плоскость. Параллельное проецирование предполагает наличие:

- центра проецирования в бесконечности,
- вектора проецирования и проецирующих лучей, параллельных данному вектору,
- проецирующей (картинной) плоскости.

При центральном проецировании явно задаются:

- координаты точки - центра проецирования,
- картинная плоскость.

Центральное проецирование порождает визуальный эффект, аналогичный зрительному - перспективное укорачивание: с увеличением расстояния от центра проекции до объекта его размеры уменьшаются. Поэтому оно используется для создания реальных картин, но непригодно для представления точной формы и размеров объектов проецирования, необходимое, например, в чертежных задачах конструкторской графики.

Параллельное проецирование дает менее реалистичное изображение (нет перспективного укорачивания), но предоставляет пользователю истинные с точностью до скалярного множителя размеры.

Для описания преобразований проецирования используются однородные координаты и матрицы четвертого порядка, что упрощает изложение материала и облегчает решение задач геометрического моделирования.

Матрицы проецирования

1. Одноточечные (имеющие одну точку схода) центральные проекции характеризуются следующим: плоскость проекции совпадает с координатной Z=0, центр проекции имеет координаты (0,0,-d). Матрица проецирования имеет вид $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Для получения проекции точки в пространстве с координатами (x,y,z,1) необходимо найти ее новые однородные, а затем - новые координаты (x', y') так

$$[xo \quad yo \quad zo \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad 0 \quad z/d+1]$$

$x' = x/(z/d+1$

$y' = y/(z/d+1))$

2. Ортографические (вид спереди, сверху, сбоку) параллельные проекции: картинная плоскость совпадает с одной из координатных, направление проецирования - с одной из главных осей.

Матрица проецирования вдоль оси X на плоскость YOZ $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

При построении вида сбоку x – координату точки проекции заменяют координатой z, y - координата остается без изменения.

Вдоль оси Y на плоскость XOZ: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

При построении вида сверху y – координату точки проекции заменяют координатой z. x - координата остается без изменения.

Вдоль оси Z на XOY: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

При построении вида спереди координаты z точек проекции отбрасываются.

3. Аксонометрические прямоугольные параллельные проекции – проецирующие прямые перпендикулярны картинной плоскости, которая не совпадает (не параллельна) ни с одной из координатных плоскостей.

Общий вид матрицы таких проекций: $\begin{bmatrix} \cos p & \sin f * \sin p & 0 & 0 \\ 0 & \cos f & 0 & 0 \\ \sin p & \sin f * \cos p & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$,

где p и f - углы, которые нормаль к картинной плоскости образует с ортами координатных осей (соответственно OY и OX).

Для построения стандартной изометрии следует взять p равным 45, f - 35.264 градусам. Для стандартной диметрии: p=22.208,f=20.705 градусов. При других значениях углов получается триметрию. Значения углов в матрицу подставляются в радианах.

4. Косоугольная параллельная аксонометрия - проекторы не перпендикулярны картинной плоскости, которая совпадает (параллельна) с одной из координатных плоскостей. Самые простые и наглядные из косоугольных - фронтальные проекции (картинная плоскость параллельна XOZ).Из них - косоугольная фронтальная диметрия (кабине) и косоугольная фронтальная изометрия (кавалье). Матрицы для этих двух проекций выглядят так:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l*\cos\pi/4 & l*\sin\pi/4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где l = 1 для кавалье и 0.5 для кабине, π=3.14159.

Для получения координат проекции любой точки изображения необходимо исходные координаты этой точки перемножить с соответствующей матрицей. Например, для получения проекции куба на экране, необходимо найти новые координаты восьми точек - вершин куба, затем соединить их отрезками в определенной последовательности. Процедура нахождения новых координат проекции кавалье, например, будет выглядеть так:

$$[z' \quad y' \quad z' \quad 1] = [z \quad y \quad z \quad 1]* \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\pi/4 & l\sin\pi/4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x + z*l\cos\pi/4 \quad y + z*l\sin\pi/4 \quad 0 \quad 1]$$

$$x' = x + z*l*0.707$$
$$y' = y + z*l*0.707.$$

В данной лабораторной работе достаточно описать 3D-объект координатами его вершин. Например, брусок с длинами сторон по X,Y,Z = (30,50,70) пикселов:

```
type
  TXYZ=record x,y,z:integer; end;

    const V:array[0..7] of TXYZ= (
  (x:-15;y: 25;z:-35),
  (x:-15;y:-25;z:-35),
  (x: 15;y:-25;z:-35),
  (x: 15;y: 25;z:-35),
  (x: 15;y: 25;z: 35),
  (x: 15;y:-25;z: 35),
  (x:-15;y:-25;z: 35),
  (x:-15;y: 25;z: 35)
  );
```

Задание на лабораторную работу:

1. Описать брусок в приборной системе координат.
2. Вывести на экран три его ортогональные проекции (вид спереди, сверху, сбоку).
3. Продемонстрировать три прямоугольные аксонометрические проекции данного бруска (изометрию, диметрию, триметрию).
4. Построить две косоугольные аксонометрические проекции бруска (кавалье, кабине).
5. Показать одноточечную центральную проекцию бруска.

Требования к защите лабораторной работы

Защита лабораторной работы состоит из демонстрации преподавателю результатов выполнения задания на лабораторную работу и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету.

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист с названием работы, оформленный согласно требованиям кафедры и распечатку программы с комментариями - результат выполнения задания.

Вопросы для самоподготовки:

1. Как из однородных координат получить реальные?
2. Чем отличаются параллельное и перспективное (центральное) проецирование? Где применяются то и другое?
3. Каким образом можно получить ортографические проекции тела на картинную плоскость?
4. Что такое аксонометрия? Какой она может быть?
5. Каким образом можно получить изометрию (диметрию) каркаса куба на экране?
6. К какой группе проекций относятся кавалье и кабине.? Чем они отличаются и как их можно получить?

Литература

1. Фоли, Дж. Основы интерактивной машинной графики [текст]: В 2 кн. кн.1, 2 / Дж. Фоли, А. Дэм; под ред. Ю. М. Баяковского; пер. с англ. В. А. Галактионова и др. - М.: Мир, 1985.
2. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] /Е.В. Шикин, А.В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.
3. Боресков, А. В. Компьютерная графика: первое знакомство [текст] / А. В. Боресков, Е. В. Шикин, Г. Е. Шикина; Под ред. Е. В. Шикина. - М.: Финансы и статистика, 1996. - 176с.: ил. - (Диалог с компьютером).

Лабораторная работа №12,13
Аффинные преобразования в пространстве

Цель работы: Закрепить лекционный материал по изучению материала одноименной темы, реализовав матрицы переноса, масштабирования, отражения и вращения применительно к координатам описанной в программе объемной фигуры (многогранника) с целью демонстрации движения и преобразования формы этой фигуры в пространстве.

Краткие теоретические сведения.

Пусть дана тройка векторов a,b,c с общим началом в точке O (начале координат), не лежащих в одной плоскости. Относительно точки O положение любой точки M в пространстве характеризуется ее радиус - вектором r. Если принять векторы a,b,c за базис и представить r=l*a+w*b+v*c, то положение точки M характеризуется набором чисел (l,w,v), называемых аффинными координатами точки M, т.е.координатами ее радиус-вектора r. Если базис (a,b,c) - единичный и взаимно перпендикулярный, то система координат - декартова. Тогда основные векторы (a,b,c) принято обозначать как i,j,k- декартов базис, координаты - (x,y,z), r=x*i+y*j+z*k.

Как и на плоскости следует применять геометрические преобразования к точкам изображения. Точка в пространстве задается либо радиус-вектором r; либо матрицами координат (x y z). Аффинные преобразования пространства определяются системой координат Oxyz, матрицей коэффициентов: $\begin{bmatrix} a & b & h \\ c & d & f \\ p & q & r \end{bmatrix}$ и числами m,n,l ,осуществляющими перенос изображения.

Если новая и старая системы координат имеют одно начало, преобразования в матричной форме выглядят так:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} * \begin{bmatrix} a & c & p \\ b & d & q \\ h & f & r \end{bmatrix} = \begin{bmatrix} xa+by+hz & cx+dy+fz & px+qy+rz \end{bmatrix}$$

В противном случае целесообразно прибегнуть к однородным координатам:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} * \begin{bmatrix} a & c & p & 0 \\ b & d & q & 0 \\ h & f & r & 0 \\ m & n & l & 1 \end{bmatrix} = \begin{bmatrix} ax+by+hz+m & cx+dy+fz+n & px+qy+rz+l & 1 \end{bmatrix}$$

Направляющие косинусы вектора - это косинусы углов, которые он образует с осями координат. Они полностью определяют направление вектора. Если вектор $a=a_x*i+a_y*j+a_z*k$, то a_x -проекция вектора a на ось x, умноженная на косинус угла между векторами и осью x. Аналогично для a_y и a_z. Если вектор a задан координатами начальной и конечной точек (x1,y1) и (x2,y2), направляющие косинусы определяются уравнениями:

$$\cos\alpha = (x2 - x1)/\sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2}$$
$$\cos\beta = (y2 - y1)/\sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2}$$
$$\cos\chi = (z2 - z1)/\sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2}$$

где α,β,γ - углы, которые вектор a образует с осями Ox, Oy, Oz, соответственно.

Частные случаи аффинных преобразований:

1.Сдвиг - перенос точки (x,y,z) на m единиц по координате x, на n - по y, на l единиц- по z.: $[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ m & n & l & 1 \end{bmatrix} = [x+m \quad y+n \quad z+l \quad 1]$

2.Поворот точки (x,y,z) вокруг оси абсцисс на угол δ:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\delta & \sin\delta & 0 \\ 0 & -\sin\delta & \cos\delta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad y\cos\delta - z\sin\delta \quad -y\sin\delta + z\cos\delta \quad 1]$$

вокруг оси ординат на угол λ:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} \cos\lambda & 0 & -\sin\lambda & 0 \\ 0 & 1 & 0 & 0 \\ \sin\lambda & 0 & \cos\lambda & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x\cos\lambda + z\sin\lambda \quad y \quad -x\sin\lambda + z\cos\lambda \quad 1]$$

вокруг оси аппликат на угол μ:

$$[z' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} \cos\mu & \sin\mu & 0 & 0 \\ -\sin\mu & \cos\mu & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x\cos\mu - y\sin\mu \quad x\sin\mu + y\cos\mu \quad z \quad 1]$$

вокруг оси, проходящей через начало координат на угол ν

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] *$$
$$\begin{bmatrix} n_1^2 + (1-n_1^2)\cos\nu & n_1 n_2 (1-\cos\nu) + n_3 \sin\nu & n_1 n_3 (1-\cos\nu) - n_2 \sin\nu & 0 \\ n_1 n_2 (1-\cos\nu) - n_3 \sin\nu & n_2^2 + (1-n_2^2)\cos\nu & n_2 n_3 (1-\cos\nu) + n_1 \sin\nu & 0 \\ n_1 n_3 (1-\cos\nu) + n_2 \sin\nu & n_2 n_3 (1-\cos\nu) - n_1 \sin\nu & n_3^2 + (1-n_3^2)\cos\nu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Здесь n_1,n_2,n_3 - направляющие косинусы вращения. Если ось вращения проходит через начало координат и точку (x_1,y_1, z_1) то

$$n_1 = x_1 / \sqrt{(x_1^2 + y_1^2 + z_1^2)}$$
$$n_2 = y_1 / \sqrt{(x_1^2 + y_1^2 + z_1^2)}$$
$$n_3 = z_1 / \sqrt{(x_1^2 + y_1^2 + z_1^2)}$$

Вращение вокруг произвольной оси осуществляется переносом ее вместе с изображением в начало координат, вращением вокруг перенесенной оси и обратным переносом изображения в исходное положение. Итоговая матрица будет определена умножением соответствующих простых матриц.

3. Симметрия относительно координатных плоскостей осуществляет зеркальное отображение 3D-изображения.

Относительно плоскости XOY

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad y \quad -z \quad 1]$$

Плоскости XOZ - $[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x \quad -y \quad z \quad 1]$ -

Плоскости YOZ : $[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [-x \quad y \quad z \quad 1]$ -

Симметрия относительно произвольной плоскости - это сложное преобразование, осуществляемое из простых поэтапно:

- совмещение плоскости симметрии с одной из координатных
- отражение относительно этой координатной плоскости
- обратное преобразование, возвращающее плоскость симметрии в исходное состояние.

4.Масштабирование осуществляется диагональными элементами матрицы преобразования.

Относительно начала координат:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x * k_x \quad y * k_y \quad z * k_z \quad 1],$$

где k_x, k_y, k_z - коэффициенты искажения вдоль осей ox,oy,oz, соответственно.

Относительно произвольного центра с координатами (m,n,l):

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] * \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ -mk_x + m & -nk_y + n & -lk_z + l & 1 \end{bmatrix} =$$

$$[xk_x - mk_x + m \quad yk_y - nk_y + n \quad zk_z - lk_z + l \quad 1]$$

Любое другое преобразование может быть представлено суперпозицией вращений, переносов, отражений, растяжений (сжатий).

Задание на лабораторную работу:

Написать на языке PASCAL программу:

1. Описывающую многогранник (куб) в приборной системе координат.
2. Смещающую его на n пикселов вправо, m - вниз, p - вглубь.
3. Зеркально отражающую относительно плоскостей координат.
4. Растягивающую (сжимающую) его вдоль координатных осей относительно некоторой заданной точки.
5. Вращающую его относительно линии, проходящей через начало координат (относительно координатных осей, диагонали многогранника).
6. Реализовать интерактивную 3Д анимацию (взаимодействующую с пользователем), на которой выполняется поворот и масштабирование объектов.

Для показа результатов работы пунктов 2-4 можно воспользоваться аксонометрической проекцией многогранника (любой по выбору), для пункта 5 задания можно использовать ортогональную проекцию.

Дополнительное задание:

1.Реализовать сложное преобразование, например, вращение с уменьшением.
2.Задание выполнить с помощью программ работы с мышью.

Требования к защите лабораторной работы

Защита работы состоит из демонстрации преподавателю результатов выполнения заданий на лабораторной работе и ответов на задаваемые преподавателем вопросы по ходу демонстрации.

Требования к отчету

Отчет выполняется на отдельных листах формата А4. Содержит титульный лист с названием работы, оформленный согласно требованиям кафедры и распечатку программы - результат выполнения заданий лабораторной работы с обязательными комментариями.

Вопросы для самоподготовки:

1. Для чего используются однородные координаты?
2. Какие преобразования считаются простыми, как выглядят матрицы таких преобразований?
3. Что такое направляющие косинусы вращения, как они определяются и где используются?
4. Каков смысл у знаков "минус" в матрицах вращения?
5. Каким образом значения коэффициентов масштабирования влияют на изображение?
6. Как осуществить любое сложное преобразование?

Литература:

1. Фоли, Дж. Основы интерактивной машинной графики [текст]: В 2 кн. кн. 1 / Дж. Фоли, А. Дэм; под ред. Ю. М. Баяковского; пер. с англ. В. А. Галактионова и др. - М.: Мир, 1985. - 367с.: ил.
2. Гилой, В. К. Интерактивная машинная графика: Структуры данных, алгоритмы, языки [текст] / В. К. Гилой; пер. с англ. под ред. Ю. М. Баяковского. - М.: Мир, 1981. - 380с.: ил.
3. Шикин, Е. В. Компьютерная графика: Динамика, реалист. изображения [текст] / Е. В. Шикин, А. В. Боресков. - М.: ДИАЛОГ-МИФИ, 1996. - 287с.: ил.
4. Боресков, А. В. Компьютерная графика: первое знакомство [текст] / А. В. Боресков, Е. В. Шикин, Г. Е. Шикина; Под ред. Е. В. Шикина. - М.: Финансы и статистика, 1996. - 176с.: ил. - (Диалог с компьютером).