Министерство образования Российской Федерации

Тамбовский Государственный Технический Университет

Кафедра: САПР

КУРСОВАЯ РАБОТА

По дисциплине: «Компьютерная графика»

Тема: «Фрактальная графика»

Выполнил: студент гр. -31

Д. Ю.

Проверил: преподаватель кафедры САПР

С. А.

Тамбов .

**Оглавление:**

[1. Задание 3](#_Toc41666354)

[2. Анализ предметной области: 3](#_Toc41666355)

[3. Описание алгоритмов. 3](#_Toc41666356)

[3. 1. Закраска по Гуро: 3](#_Toc41666357)

[3. 2. Точное текстурирование: 4](#_Toc41666358)

[3. 3. Алгоритм удаления невидимых граней 7](#_Toc41666359)

[3. 3. 1. Алгоритм построчного сканирования 7](#_Toc41666360)

[3. 3. 2. Алгоритм построчного сканирования, использующий Z-буфер. 8](#_Toc41666361)

[4. Описание программы. 10](#_Toc41666362)

[5. Описание программы реализующей алгоритм: 11](#_Toc41666363)

[6. Выводы 13](#_Toc41666364)

[Литература: 13](#_Toc41666365)

[Текст программных модулей. 14](#_Toc41666366)

1. Задание

Реализовать алгоритм удаления скрытых поверхностей по методу «Построчного сканирования с использование Z-буффера».

При этом работа должна содержать:

Раскраску объектов сцены диффузной моделью освещения.

Работу с текстурами.

3D- преобразования.

2. Анализ предметной области:

Построение 3D-изображений в компьютерной графике одна из важнейших задач требующая большого количества вычислений в связи с чем и было разработано множество различных алгоритмов (удовлетворяющих требованиям качества построения изображения и скорости обработки для той или иной задачи).

Для выполнения задания будем использовать интервальный алгоритм «Построчного сканирования с использование Z-буффера» для удаления невидимых граней.

Для освещения сцены взят один источник освещения, центральная проекция и точное текстурирование.

Эти алгоритмы обеспечат приемлимое качество изображения и достаточную скорость.

3. Описание алгоритмов.

**3. 1. Закраска по Гуро:**

Т.к. используется диффузное освещение и в сцене участвует несколько источников света, то каждый источник будет вносить свой вклад в освещение грани. Поэтому необходимо вычислить и сложить все значения освещённости от каждого источика.

Основная идея закраски по Гуро заключаеться в линейной интерполяции освещённости в точке грани. Для этого вычислим значения освещённости в вершинах каждой грани. Для чего вычисляем векторное произведение вектора освещения и вектора нормали для каждой вершины треугольников. Так как вектора при перемножении приведены к единичной длине ( нормализованы ), то полученноое произведение численно равно косинусу угла между нормалью в вершине и вектором освещения. Освещённость в вершине прямо пропорциональна значению косинуса угла. Получив освещённость в вершине будем линейно интерполировать её по гарни и использовать для коррекции освещённости при построении треугольника.

**3. 2. Точное текстурирование:**

Задача текстурирования формулируется таким образом: есть грань - согласно предположениям, треугольная - с наложенной на нее текстурой. То есть каждая точка грани окрашена цветом соответствующей ей точки в текстуре. Текстура накладывается линейным образом. Есть точка экрана с координатами на экране (sx,sy), принадлежащая проекции грани. Требуется найти ее цвет, то есть цвет соответствующей этой точке экрана точки текстуры. А для этого надо найти координаты текстуры для этой точки - точнее, для той точки, проекцией которой на экран является наша (sx,sy).

Пусть вершины грани есть точки A(Ax,Ay,Az), B(Bx,By,Bz) и C(Cx,Cy,Cz), а соответствующие им точки текстуры - (Au,Av), (Bu,Bv) и (Cu,Cv). Найдем координаты текстуры для точки, проекцией которой является (sx,sy).

Для точек (x,y,z), проекцией которых является (sx,sy) имеем:

sx = xSize/2+x\*dist/(z+dist),

sy = ySize/2-y\*dist/(z+dist).

Для упрощения формул будем использовать обозначения

i = sx-XSize/2,

j = YSize/2-sy,

Z = z+dist.

Тогда эти формулы примут вид

i = x\*dist/Z,

j = y\*dist/Z,

или, что равносильно:

i\*Z = x\*dist,

j\*Z = y\*dist.

Рассмотрим точку D, принадлежащую грани. Для нее D = A+a\*(B-A)+b\*(C-A), так как она лежит в грани. D однозначно задается парой (a,b). Для нее координаты текстуры (из того, что текстура накладывается линейно) будут такие:

Du = Au+a\*(Bu-Au)+b\*(Cu-Au),

Dv = Av+a\*(Bv-Av)+b\*(Cv-Av).

Пусть проекция D на экран как раз и имеет координаты (sx,sy), тогда для нее выполнены написанные выше соотношения:

i\*DZ = Dx\*dist,

j\*DZ = Dy\*dist.

Подставим сюда координаты D, выраженные через координаты A, B и неизвестные коэффициенты a, b:

i\*(Az+a\*(Bz-Az)+b\*(Cz-Az)+dist) = dist\*(Ax+a\*(Bx-Ax)+b\*(Cx-Ax)),

j\*(Az+a\*(Bz-Az)+b\*(Cz-Az)+dist) = dist\*(Ay+a\*(By-Ay)+b\*(Cy-Ay)),

т.к. мы договорились, что AZ = Az+dist, и, кстати, поэтому BZ-AZ = Bz-Az,это можно чуть укоротить:

i\*(AZ+a\*(BZ-AZ)+b\*(CZ-AZ)) = dist\*(Ax+a\*(Bx-Ax)+b\*(Cx-Ax)),

j\*(AZ+a\*(BZ-AZ)+b\*(CZ-AZ)) = dist\*(Ay+a\*(By-Ay)+b\*(Cy-Ay)).

Выделим a и b:

a\*(i\*(BZ-AZ)-dist\*(Bx-Ax))+b\*(i\*(CZ-AZ)-dist\*(Cx-Ax)) = dist\*Ax-i\*AZ,

a\*(j\*(BZ-AZ)-dist\*(By-Ay))+b\*(j\*(CZ-AZ)-dist\*(Cy-Ay)) = dist\*Ay-j\*AZ.

Получили систему двух линейных уравнений, из которой можно найти a и b, а по ним немедленно вычисляются u и v.

Введем вектор M = BA (Mx = Bx-Ax и т.д.) и вектор N = CA, обозначим d = dist (все это для краткости записи).

Тогда эти два уравнения перепишутся в виде:

a\*(i\*Mz-d\*Mx)+b\*(i\*Nz-d\*Nx) = d\*Ax-i\*AZ,

a\*(j\*Mz-d\*My)+b\*(j\*Nz-d\*Ny) = d\*Ay-j\*AZ,

и решая систему (например, по формулам Крамера) получаем:

i(AZNy-AyNz)+j(AxNz-AZNx)+d(AyNx-AxNy)

a = --------------------------------------------------------------

i(MyNz-MzNy)+j(NxMz-MxNz)+d(MxNy-NxMy)

аналогично получаем

i(AZMy-AyMz)+j(AxMz-AZMx)+d(AyMx-AxMy)

b = ------------------------------------------------------------

i(MyNz-MzNy)+j(NxMz-MxNz)+d(MxNy-NxMy)

Знаки умножения здесь везде опущены для удобства записи формул.

Подставить Mx = Bx-Ax и так далее, а потом подставить эти a и b в формулы для Du и Dv. В процессе подстановок что-то сократится.

**3. 3. Алгоритм удаления невидимых граней**

*3. 3. 1. Алгоритм построчного сканирования*

Алгоритмы Варнока, *z*-буфера и строящий список приоритетов обрабатывают элементы сцены или многоугольники в порядке, который не связан с процессом визуализации. В то же время алгоритмы построчного сканирования, впервые предложенные Уайли и др. , Букнайтом и Уоткинсом , обрабатывают сцену в порядке прохождения сканирующей прямой. Эти алгоритмы оперируют в пространстве изображения .

Алгоритмы удаления невидимых линий и поверхностей с пoстрочным, сканированием являются обобщением методов растровой развертки многоугольников. Эти алгоритмы сводят трехмерную задачу удаления невидимых линий и поверхностей к двумерной. Сканирующая плоскость определяется точкой наблюдения, расположенной в бесконечности на положительной полуоси Z, и сканирующей строкой так, как показано на рис. 1. Пересечение сканирующей плоскости и трехмерной сцены определяет окно размером в одну сканирующую строку. Задача удаления невидимых поверхностей решается в пределах этого окна, образованного сканирующей плоскостью.

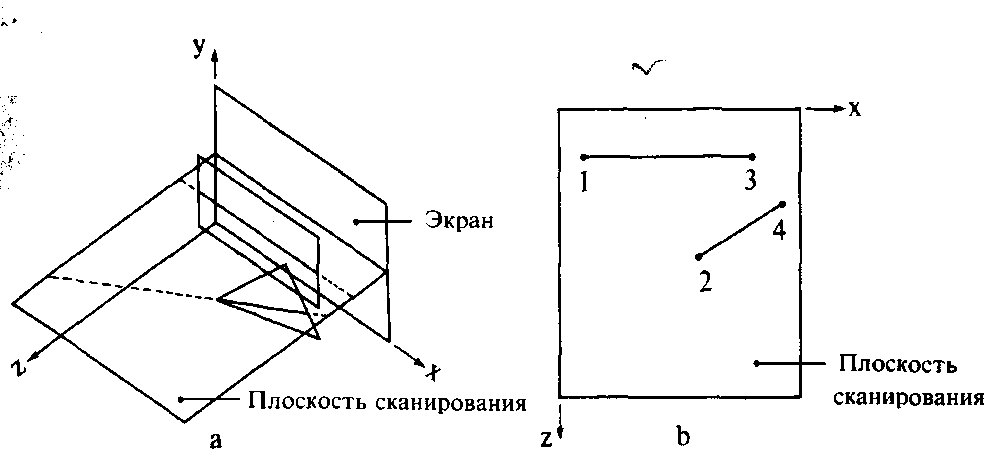


Рис. 1. Сканирующая плоскость.

На рис. 1, b представлено пересечение сканирующей плоскости с многоугольниками. Этот рисунок показывает, что задача удаления невидимых поверхностей сводится к задаче о том, какой отрезок видим для каждой точки сканирующей строки. На первый взгляд может показаться, что здесь можно непосредственно применить алгоритм формирования упорядоченного списка ребер. Однако, из рис. 1, b ясно видно, что это приведет к некорректным результатам. Например, для сканирующей строки, показанной на рис. 1, существуют четыре активных ребра в списке активных ребер. Пересечения этих ребер со сканирующей строкой показаны точками на рис. 1, b; список упорядоченных ребер обозначен цифрами. Попарная выборка пересечений приводит к тому, что пикселы между точками 1 и 2, а также между точками 3 и 4 активируются. В то же время пикселы между точками 2 и 3 не активируются. Таким образом, получается неверный результат. «Дыра» образовалась в том месте, где сканирующая строка фактически пересекла два многоугольника.

*3. 3. 2. Алгоритм построчного сканирования, использующий Z-буфер.*

Одним из простейших алгоритмов построчного сканирования, который решает задачу удаления невидимых поверхностей, является специальный случай алгоритма z-буфера. Будем называть его алгоритмом построчного cканирования с Z-буфером . Используемое в этом алгоритме окно визуализации имеет высоту в одну сканирующую строку и ширину во весь экран. Как для буфера кадра, так и для Z-буфера требуется память высотой в 1 бит, шириной в горизонтальное разрешение экрана и глубиной в зависимости от требуемой точности. Обеспечиваемая точность по глубине зависит от диапазона значений, которые может принимать *z.* Например, буфер кадра может иметь размер 1х512х24 бит, а Z-буфер — 1х512х20 бит.

Концептуально этот алгоритм достаточно прост. Для каждой сканирующей строки буфер кадра инициализируется с фоновым значением интенсивности, а Z-буфер — с минимальным значением Z*.* Затем определяется пересечение сканирующей строки с двумерной проекцией каждого многоугольника сцены, если они существуют. Эти пересечения образуют пары.

При рассмотрении каждого пиксела на сканирующей строке в интервале между концами пар его глубина сравнивается с глубиной, содержащейся в соответствующем элементе Z-буфера. Если глубина этого пиксела больше глубины из Z-буфера, то рассматриваемый отрезок будет текущим видимым отрезком. И следовательно, атрибуты многоугольника, соответствующего данному отрезку, заносятся в буфер кадра в позиции данного пиксела; соответственно корректируется и Z-буфер в этой позиции. После обработки всех многоугольников сцены буфер кадра размером в одну сканирующую строку содержит решение задачи удаления невидимых поверхностей для данной сканирующей строки. Он выводится на экран дисплея в порядке, определяемом растровым сканированием, т.е. слева направо. В этом   
алгоритме можно использовать методы устранения ступенчатости, основывающиеся как на пре, так и на постфильтрации.

Однако практически сравнение каждого многоугольника с каждой сканирующей строкой оказывается неэффективным. Поэтому используется некоторая разновидность списка упорядоченных ребер. В частности, для повышения эффективности этого алгоритма применяются групповая сортировка по оси *у,* список активных многоугольников и список активных ребер.

4. Описание программы.

**4. 1.Общие сведения.**

Программа написана на языке Borland Pascal версии 7.1 с использованием графической библиотеки “ASY VBE” версии 1.2.

Программа работает в “HighColor” графическом режиме работы видеокарты с разрешением 640x480x16. При работе используется ( при возможности ) расширение VesaBIOS версии 2.0.

При работе программы под управлением операционной системы Windows возможна работа только в режиме FullScreen отображения окна приложения. Техническая документация написана с использованием программных продуктов Microsoft Word 2000.

**4. 2. Функциональное назначение.**

Данная программа является курсовым заданием по специальности «Машинная графика и графическое моделирование»

**4. 3. Описание логической структуры.**

Описание логической структуры приведено выше.

**4. 4. Используемые технические средства.**

Программа была написана и откомпилирована на ЭВМ Celeron 450/4.3/156/SVGA, оснащенной устройством печати CANON BJC-4650.

**5. Вызов и загрузка.**

Программа хранится в файле mg\_and\_gm.pas в каталоге d:\BorlandPascal\bin Программа вызывается на выполнение командой mg\_and\_gm.exe.

**6. Входные данные.**

Входными данными являются кнопки управления, описанные ниже, и рисунок фона и текстуры.

**7. Выходные данные.**

Выходными данными является графическое 3D изображение, иллюстрирующее 3D преобразование и источники освещения.

5. Описание программы реализующей алгоритм:

Программа состоит из набора процедур, используемых для обработки и построения объекта, и основного модуля управления, используемого для организации интерактивного диалога с пользователем.

**Процедуры можно разделить на группы:**

3D преобразования:  
Move, Rot, Scale, Mult.

Вывод объектов:  
DrawScene, TransTringle, SortTriangle, CreatLineList,SortLine, CreatInterval, SortInterval, Watkins.

Чтение данных из файлов:  
ReadObject.

Дополнительные:  
PtogramInit, ActivateScreen, ProgramDone.

**Описание формата данных 3D модели:**

Для описания 3D модели в программе используются несколько файлов.

**1.** Для описания объектов сцены используется текстовый файл ”\*.3D” - текстовый файл, который описывает основные данные о вершинах и гранях объекта:

Количество вершин объекта.

Список вершин объекта и соответствие им координат в текстуре.

Колличество граней объекта.

Список граней объекта.

**2.** Для описания 3D модели используется типизированный файл содержащий список треугольных граней записанных в формате:

**OVertex**=Record

x,y,z : Real; {координаты}

tx,ty : Integer; {координаты в текстуре}

nx,ny,nz: Real; {направляющие косинусы нормали}

End;

**OTriangle**=Record

A,B,C: OVertex;

End;

**3.** Для текстурирования объекта используются 8 битные текстуры произволного размера. Текстуры считываются программой из графического файла формата BMP.

**Описание пользовательского интерфейса**:

Для интерактивного управления объектом сцены и источником освещения используются клавиши:

|  |  |
| --- | --- |
| SiftUp | Поворот вокруг оси X |
| SiftDown | Поворот вокруг оси X |
| SiftLeft | Поворот вокруг оси Y |
| SiftRight | Поворот вокруг оси Y |
| CtrlLeft | Поворот вокруг оси Z |
| CtrlRight | Поворот вокруг оси Z |
| PageUp | Увеличение |
| PageDoun | Уменьшение |
| Right | Перемещения вдоль оси X |
| Left | Перемещения вдоль оси X |
| Up | Перемещения вдоль оси Y |
| Down | Перемещения вдоль оси Y |
| GreyPlus | Сдвиг источника освещения |
| GreyMinu | Сдвиг источника освещения |
| Home | Переход в начальное положение |
| Esc | Выход из программы |

6. Выводы

Построение реалистичной 3D-сцены требует больших вычислительных ресурсов, путём усложнения вычислений можно добиться высокого качества изображения теряя в скорости именно поэтому многие алгоритмы 3D графики переносяться на аппаратную основу. (Закраска, диффузное освещение, 3D-преобразования, Удаление невидимых граней, текстурирование );

Литература:

1. Алгоритмические основы машинной графики. Дэвид Роджерс. 1989г.

2. Геометрическое моделирование и машинная графика в САПР. Киев 1991г.

3. Интерактивная машинная графика. Тамбов 1993.

4. Компьютерная графика динамика, реалистические изображения.  
Шикин Е.В.,Боресков А.В., Диалог-МИФИ

Текст программных модулей.

Program \_3D\_;

{$M 16384,0,$100}

Uses nVBE,hDriver,Crt,nAccess ;

{$I TOOLS.INC}

{$I COLORRGB.INC}

{$I MATRIX.INC}

{$R+,Q+}

{---------------------------------------------------------------------------}

Type

Vector=Record x,y,z : Real; End;

OVertex=Record

x,y,z : Real; {координаты}

tx,ty : Integer; {координаты в текстуре}

nx,ny,nz: Real; {направляющие косинусы вектора нормали}

End;

MVertex=Record

{координаты}

x,y,z: Real;

tx,ty: Integer; {координаты в текстуре}

c : Real; {освещенность}

End;

OTriangle=Record

A,B,C: OVertex;

End;

MTriangle=Record

A,B,C: MVertex;

End;

LPoint=Record

x,z: Real;

tx,ty: Integer;

c: Real;

End;

SLine=Record

A,B: LPoint;

End;

Interval=Record

n: Integer;

x,z: Real;

End;

Const

TCount : Integer=0; {счетчик треугольников}

MTCount: Integer=0; {счетчик треугольников}

LCount : Integer=0; {счетчик линий сканирования}

ICount : Integer=0; {счетчик интервалов}

y : Integer=0; {счетчик линий сканирования}

LV1 : Vector=(x: 0; y:0; z:2);{вектор освещения}

LV2 : Vector=(x: 0; y:-1; z:0);{вектор освещения}

n = 50;

Texture: String='cat.bmp';

fone : LongInt=0;

XRes = 350;

YRes = 250;

Var

T: Array[0..n] Of OTriangle;

MT: Array[0..n] Of MTriangle;

L: Array[0..n] Of SLine;

ZBuf: Array[0..XRes] Of Real;

Screen,TBuf: PBitmap16;

H : Integer;

Tex,Logo,WW : PBitmap;

R,M,Res : Matrica;

Key : Word;

Procedure fntFONT;external;{$L FONT.obj}

Procedure ActivateScreen(Mode:Word);

begin

vActivateDriver(@vHighDriver);

vSetMode(Mode);

end;

Procedure Out\_Text(Str:String; y: Integer);

Var x,i,N: Integer;

Begin

vSetFont(@fntFont,8,1);

N:=Length(Str);

x:=320-N\*10;

vSetColor(blue);

For i:=1 to N Do

Begin

vTextAsSprite(x,y,Str[i]);

Inc(x,10);

Delay(800);

End;

End;

Procedure ProgramInit;

Var i,j: Integer; c: LongInt;

Begin

ActivateScreen(vDetectMode(640,480,16));

vSetColor(0); vClearScreen;

{-----------------------------------}

vLoadFromBMP('fon.bmp',WW,Palette);

For j:=0 To 479 Do

For i:=0 To 639 Do

Begin

C:=WW^.GetPixel(i,j);

C:=vColor((Palette[C].r shl 2), (Palette[C].g shl 2), (Palette[C].b shl 2));

vSetColor(c);

vPutPixel(i,j);

End;

Dispose(WW,Done);

vSetColor(65535);

Out\_Text('КУРСОВАЯ РАБОТА ПО МГ И ГМ',320);

Out\_Text('Выполнил студент гр.В-41',350);

Out\_Text('Садовов Андрей',380);

TBuf := New(PBitmap16,Init(XRes,YRes));

TBuf^.GetFromScreen(50,30);

vLoadFromBMP(Texture,Tex,Palette);

For i:=0 To 255 Do

Begin

Palette[i].R:=Palette[i].R shl 2;

Palette[i].G:=Palette[i].G shl 2;

Palette[i].B:=Palette[i].B shl 2;

End;

Screen := New(PBitmap16,Init(XRes,YRes));

End;

Procedure ProgramDone;

Begin

Dispose(Screen,Done);

Dispose(Tex,Done);

vCloseGraph;

vReleaseDriver(vCurrentDriver);

End;

Procedure ReadObject(FileName:String);

Var F: File of OTriangle;

Begin

Assign(F,FileName);

Reset(F);

While not EOF(F) Do

Begin

Read(F,T[TCount]);

Inc(TCount);

End;

Close(F);

End;

Procedure GuroZLine(Buf: PBitmap16; Ax,Az,Bx,Bz: Real; y: Integer; Atx,Aty,Btx,Bty: Real;

Ac, Bc : Real);

Var

I,x : Integer;

L: Real;

dC,C : Real;

z,dz: Real;

dtx,dty: Real;

tx,ty: Real;

Color: LongInt;

Begin

L := Bx-Ax+1;

dC := (Bc-Ac)/L;

dz := (Bz-Az)/L;

dtx := (Btx-Atx)/L;

dty := (Bty-Aty)/L;

tx:=Atx;

ty:=Aty;

C:=Ac;

z:=Az;

For i := Round(Ax) To Round(Bx) Do

Begin

C:=C+dC;

x:=i;

z:=z+dz;

tx:=tx+dtx;

ty:=ty+dty;

If (x<0)or(x>xres) Then Continue;

If z>ZBuf[x] Then Continue

Else ZBuf[x]:=z;

Color:=Tex^.GetPixel(Round(tx),Round(ty));

Color:=vColor(Round((Palette[Color].r)\*c),

Round((Palette[Color].g)\*c),

Round((Palette[Color].b)\*c));

Buf^.PutPixel(x,y,Color);

End;

End;

Procedure SortTriangle;

Var i: Integer;

t: MVertex;

Begin

{ Сортировка вершин по возростанию координаты `y` }

For i:=0 To MTCount-1 Do

Begin

With MT[i] Do

Begin

If A.y>B.y Then Begin t:=A; A:=B; B:=t; End;

If B.y>C.y Then Begin t:=B; B:=C; C:=t; End;

If A.y>B.y Then Begin t:=A; A:=B; B:=t; End;

End;

End;

End;

Procedure SortLine;

Var i: Integer;

t: LPoint;

Begin

{ Сортировка концов отрезков по возростанию координаты `x` }

For i:=0 To LCount-1 Do

If L[i].A.x>L[i].B.x Then

Begin t:=L[i].A; L[i].A:=L[i].B; L[i].B:=t; End;

End;

Procedure TransTriangle(M: Matrica);

Var nx,ny,nz,NLen,znormal,VectCos,VectCos1,VectCos2: Real; i: Integer;

Begin

MTCount:=0;

For i:=0 To TCount-1 Do

Begin

MT[MTCount].A.x:=T[i].A.x\*M[1,1] + T[i].A.y\*M[1,2] + T[i].A.z\*M[1,3] +M[1,4];

MT[MTCount].A.y:=T[i].A.x\*M[2,1] + T[i].A.y\*M[2,2] + T[i].A.z\*M[2,3] +M[2,4];

MT[MTCount].A.z:=T[i].A.x\*M[3,1] + T[i].A.y\*M[3,2] + T[i].A.z\*M[3,3] +M[3,4];

{---}

MT[MTCount].B.x:=T[i].B.x\*M[1,1] + T[i].B.y\*M[1,2] + T[i].B.z\*M[1,3] +M[1,4];

MT[MTCount].B.y:=T[i].B.x\*M[2,1] + T[i].B.y\*M[2,2] + T[i].B.z\*M[2,3] +M[2,4];

MT[MTCount].B.z:=T[i].B.x\*M[3,1] + T[i].B.y\*M[3,2] + T[i].B.z\*M[3,3] +M[3,4];

{---}

MT[MTCount].C.x:=T[i].C.x\*M[1,1] + T[i].C.y\*M[1,2] + T[i].C.z\*M[1,3] +M[1,4];

MT[MTCount].C.y:=T[i].C.x\*M[2,1] + T[i].C.y\*M[2,2] + T[i].C.z\*M[2,3] +M[2,4];

MT[MTCount].C.z:=T[i].C.x\*M[3,1] + T[i].C.y\*M[3,2] + T[i].C.z\*M[3,3] +M[3,4];

{ОТБРАСЫВАНИЕ НЕЛИЦЕВЫХ ГРАНЕЙ}

znormal:=(MT[MTCount].B.x-MT[MTCount].A.x)\*(MT[MTCount].C.y-MT[MTCount].A.y)-

(MT[MTCount].B.y-MT[MTCount].A.y)\*(MT[MTCount].C.x-MT[MTCount].A.x);

If znormal <= 0 Then Continue;

{---}

MT[MTCount].A.tx:=T[i].A.tx; MT[MTCount].A.ty:=T[i].A.ty;

MT[MTCount].B.tx:=T[i].B.tx; MT[MTCount].B.ty:=T[i].B.ty;

MT[MTCount].C.tx:=T[i].C.tx; MT[MTCount].C.ty:=T[i].C.ty;

{A}

nx:=T[i].A.nx\*M[1,1] + T[i].A.ny\*M[1,2] + T[i].A.nz\*M[1,3];

ny:=T[i].A.nx\*M[2,1] + T[i].A.ny\*M[2,2] + T[i].A.nz\*M[2,3];

nz:=T[i].A.nx\*M[3,1] + T[i].A.ny\*M[3,2] + T[i].A.nz\*M[3,3];

NLen:=Sqrt( Sqr(nx)+ Sqr(ny)+ Sqr(nz) );

if NLen<>0 Then Begin nx:=nx/NLen; ny:=ny/NLen; nz:=nz/NLen; End;

VectCos1:=nx\*LV1.x + ny\*LV1.y + nz\*LV1.z;

VectCos2:=nx\*LV2.x + ny\*LV2.y + nz\*LV2.z;

VectCos:=VectCos1+VectCos2;

If VEctCos>1 Then VectCos:=1;

If VEctCos<0 Then VectCos:=0;

MT[MTCount].A.c:=VectCos;

{B}

nx:=T[i].B.nx\*M[1,1] + T[i].B.ny\*M[1,2] + T[i].B.nz\*M[1,3];

ny:=T[i].B.nx\*M[2,1] + T[i].B.ny\*M[2,2] + T[i].B.nz\*M[2,3];

nz:=T[i].B.nx\*M[3,1] + T[i].B.ny\*M[3,2] + T[i].B.nz\*M[3,3];

NLen:=Sqrt( Sqr(nx)+ Sqr(ny)+ Sqr(nz) );

if NLen<>0 Then Begin nx:=nx/NLen; ny:=ny/NLen; nz:=nz/NLen; End;

VectCos1:=nx\*LV1.x + ny\*LV1.y + nz\*LV1.z;

VectCos2:=nx\*LV2.x + ny\*LV2.y + nz\*LV2.z;

VectCos:=VectCos1+VectCos2;

If VEctCos>1 Then VectCos:=1;

If VEctCos<0 Then VectCos:=0;

MT[MTCount].B.c:=VectCos;

{C}

nx:=T[i].C.nx\*M[1,1] + T[i].C.ny\*M[1,2] + T[i].C.nz\*M[1,3];

ny:=T[i].C.nx\*M[2,1] + T[i].C.ny\*M[2,2] + T[i].C.nz\*M[2,3];

nz:=T[i].C.nx\*M[3,1] + T[i].C.ny\*M[3,2] + T[i].C.nz\*M[3,3];

NLen:=Sqrt( Sqr(nx)+ Sqr(ny)+ Sqr(nz) );

if NLen<>0 Then Begin nx:=nx/NLen; ny:=ny/NLen; nz:=nz/NLen; End;

VectCos1:=nx\*LV1.x + ny\*LV1.y + nz\*LV1.z;

VectCos2:=nx\*LV2.x + ny\*LV2.y + nz\*LV2.z;

VectCos:=VectCos1+VectCos2;

If VEctCos>1 Then VectCos:=1;

If VEctCos<0 Then VectCos:=0;

MT[MTCount].C.c:=VectCos;

inc(MTCount);

End;

End;

Procedure CreatLineList;

Var i: Integer;

x1,x2,z1,z2,L1,L2: Real;

tx1,ty1,tx2,ty2: Integer;

Begin

LCount:=0;

For i:=0 To MTCount-1 Do

Begin

If (y<MT[i].A.y) or (y>MT[i].C.y) Then Continue;

With MT[i] Do

Begin

x1:=A.x+(y-A.y)\*(C.x-A.x) /(C.y-A.y);

z1:=A.z+(y-A.y)\*(C.z-A.z)/(C.y-A.y);

L1:=A.c+(y-A.y)\*(C.c-A.c) /(C.y-A.y);

tx1:=Round(A.tx+(y-A.y)\*(C.tx-A.tx) / (C.y-A.y));

ty1:=Round(A.ty+(y-A.y)\*(C.ty-A.ty) / (C.y-A.y));

If y<B.y Then

Begin

x2:=A.x+(y-A.y)\*(B.x-A.x) / (B.y-A.y);

z2:=A.z+(y-A.y)\*(B.z-A.z)/(B.y-A.y);

L2:=A.c+(y-A.y)\*(B.c-A.c) / (B.y-A.y);

tx2:=Round(A.tx+(y-A.y)\*(B.tx-A.tx) / (B.y-A.y));

ty2:=Round(A.ty+(y-A.y)\*(B.ty-A.ty) / (B.y-A.y));

End

else

If C.y=B.y Then

Begin

x2:=B.x; z2:=B.z; L2:=B.c; tx2:=B.tx; ty2:=B.ty;

End

else

Begin

x2:=B.x+(y-B.y)\*(C.x-B.x) / (C.y-B.y);

z2:=B.z+(y-B.y)\*(C.z-B.z)/(C.y-B.y);

L2:=B.c+(y-B.y)\*(C.c-B.c) / (C.y-B.y);

tx2:=Round(B.tx+(y-B.y)\*(C.tx-B.tx) / (C.y-B.y));

ty2:=Round(B.ty+(y-B.y)\*(C.ty-B.ty) / (C.y-B.y));

End;

L[LCount].A.x:=Round(x1); L[LCount].A.z:=z1;

L[LCount].A.tx:=tx1; L[LCount].A.ty:=ty1; L[LCount].A.c:=L1;

L[LCount].B.x:=Round(x2); L[LCount].B.z:=z2;

L[LCount].B.tx:=tx2; L[LCount].B.ty:=ty2; L[LCount].B.c:=L2;

Inc(LCount);

End;

End;

End;

Procedure LZBuf(y: Integer);

var i: Integer;

Begin

For i:=0 To XRes Do ZBuf[i]:=876786756;

For i:=0 To LCount-1 Do

With L[i] Do

GuroZLine(Screen,A.x,A.z, B.x,B.z, y, A.tx,A.ty, B.tx,B.ty, A.c,B.c);

End;

Procedure DrawScene(M: Matrica);

Begin

TransTriangle(Res);

SortTriangle;

Screen^.Fill(fone);

TBuf^.StampToBitmap(Screen,0,0);

For y:=0 To YRes Do

Begin

CreatLineList;

If LCount>0 Then

Begin

SortLine;

LZBuf(y);

End;

End;

Screen^.Stamp(50,30);

End;

Var alfa: Integer;

BEGIN

If ParamCount>0 Then Texture:=ParamStr(1);

ProgramInit;

ReadObject('Obj.3d');

H:=0;

alfa:=0;

M:=Ed; R:=Ed;

Repeat

If KeyHit Then

Begin

Key:=KeyGet;

If ShiftPressed Then

case Key of

kbRight: Rot(R,'y',355);

kbLeft: Rot(R,'y',5);

kbUp: Rot(R,'x',5);

kbDown: Rot(R,'x',355);

end

else

case Key of

kbEsc: Break;

kbLeft: Move(M,-15,0,0);

kbRight: Move(M,15,0,0);

kbUp: Move(M,0,-15,0);

kbDown: Move(M,0,15,0);

kbPgUp: Scale(M,1.2,1.2,1.2);

kbPgDn: Scale(M,1/1.2,1/1.2,1/1.2);

kbCtrlRight: Rot(R,'z',5);

kbCtrlLeft: Rot(R,'z',355);

kbGrayPlus: Begin

If LV1.z<5.9 Then LV1.z:=LV1.z+0.1;

If LV2.z<5.9 Then LV2.z:=LV2.z+0.1;

End;

kbGrayMinus: Begin

If LV1.z>0.1 Then LV1.z:=LV1.z-0.1;

If LV2.z<5.9 Then LV2.z:=LV2.z-0.1;

End;

end;

End;

Mult(M,R,Res);

Move(Res,100,80,100);

DrawScene(Res);

alfa:=(alfa +5) mod 65535;

Until False;

ProgramDone;

END.