Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИТ

Кафедра ИиВД

Специальность: 1-98 01 03 – «Программное обеспечение информационной безопасности мобильных систем»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КУРСОВОЙ РАБОТЫ

по дисциплине Компьютерная геометрия и графика

Тема Приложение Windows «Эллипсоид»

Исполнитель

студент 3 курса группы 8 А. А. Гиль

Руководитель

доцент, к. т. н. А. А. Дятко

Курсовой проект защищен с оценкой

Руководитель А. А. Дятко

Минск 2021

Оглавление

[ЗАДАНИЕ 3](#_Toc89855887)

[ВВЕДЕНИЕ 4](#_Toc89855888)

[ОБЩАЯ СХЕМА ПРИЛОЖЕНИЯ 8](#_Toc89855889)

[МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ АЛГОРИТМОВ 11](#_Toc89855890)

[ПОЯСНЕНИЯ К ЛИСТИНГАМ ПРОГРАММЫ 14](#_Toc89855891)

[ЗАКЛЮЧЕНИЕ 15](#_Toc89855892)

[ПРИЛОЖЕНИЕ 1 16](#_Toc89855893)

[ПРИЛОЖЕНИЕ 2 18](#_Toc89855894)

[ПРИЛОЖЕНИЕ 3 18](#_Toc89855895)

[СПИСОК ЛИТЕРАТУРЫ 19](#_Toc89855896)

# ЗАДАНИЕ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Построить 3D – изображение эллипсоида (Рис. 1)   |  |  | | --- | --- | |  |  |   Рис.1 Эллипсоид  Уравнение эллипсоида   |  |  |  | | --- | --- | --- | |  |  |  |   Для описания поверхности эллипсоида использовать полигональную модель.  Изменяемые параметры:   * значение полуосей эллипсоида ; * положение наблюдателя; * положение источника света; * цвет источника света; * модель освещения.   Для установки параметров модели использовать окно диалога.  Обеспечить запись изображения в графический файл \*.bmp |

# ВВЕДЕНИЕ

Компьютерная или машинная графика — это вполне самостоятельная область человеческой деятельности со своими проблемами и спецификой. Компьютерная графика — это и новые эффективные технические средства для проектировщиков, конструкторов и исследователей, и программные системы и машинные языки, и новые научные, учебные дисциплины, родившиеся на базе синтеза таких наук как аналитическая, прикладная, начертательная геометрия, программирование для ЭВМ, методы вычислительной математики и т. п. Основными задачами машинной графики являются ввод (считывание) графической информации в ЭВМ, вывод ее из ЭВМ (формирование изображений), а также определенного рода переработка информации в компьютере. Таким образом, основные задачи машинной геометрии или, как говорят, автоматизированного геометрического моделирования и конструирования — синтез в ЭВМ и анализ геометрических объектов, решение задач геометрического характера.

Разработав методы точного аналитического описания кривых и поверхностей, мы можем решать задачу об их синтезе, а именно: задачу о построении кривой или поверхности по заданным условиям. Эти условия могут быть определены точно (кривая или поверхность должна проходить через заданные опорные точки, иметь непрерывную кривизну и т. п.) или иметь вид эстетических требований, таких как красота и другие неформализуемые свойства. Нетрудно представить, какое большое значение имеет и может иметь машинная графика практически в любой области человеческой деятельности, начиная от технического проектирования и заканчивая искусством «для избранных».

Достоинства графики:

* Наиболее естественные средства общения с ЭВМ;
* Хорошо развитый двухмерный и трехмерный механизм распознавания образов позволяет очень быстро и эффективно воспринимать, и обрабатывать различные виды данных. Как гласит старинная китайская пословица: "Одна картинка стоит 1000 слов";

Главной задачей данной курсовой работы является разработка приложения для построения 3D изображения поверхности второго порядка - эллипсоида.

Поверхностью второго порядка - это геометрическая фигура, которая находится в некоторой декартовой системе координат описывается уравнением:

|  |
| --- |
|  |

Эллипсоид - это поверхность второго порядка в трёхмерном пространстве, которую мы получаем при деформации сферы вдоль трёх взаимно перпендикулярных осей.

Каноническое уравнение эллипсоида в декартовой системе координат будет выглядеть таким образом:

|  |
| --- |
|  |

Где соответственно a, b, c – произвольные положительные числа, которые являются полуосями эллипсоида.

Для x, y, z параметрическое уравнение эллипсоида, где:

|  |  |
| --- | --- |
|  |  |

Нужно отметить, что если полуоси a, b, c будут равны то мы получим ничто иное как сферу.

Если все оси эллипсоида различны, его называют трёхосным, а если какие-либо две оси эллипсоида одинаковы, то такой эллипсоид называют сфероидом.

Также, в нашем приложении эллипсоид имеет две модели отражения света от поверхности: диффузное и зеркальное. (Рисунок 1.1, 1.2)

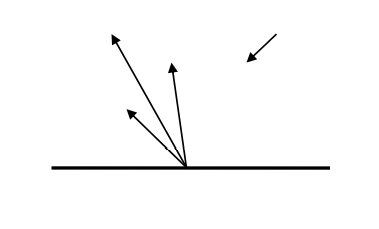


Рисунок 1.1 – Зеркальная модель отражения света

Как мы знаем из курса физики световые лучи, падающие на поверхность, могут быть поглощены, отражены или пропущены. Объект (в нашем случае поверхность эллипсоид) можно увидеть, только если он отражает и поглощает свет. Следовательно, мы будем использовать две модели освещения: зеркальную модель, и диффузионную. При зеркальной модели, отражение происходит от внешней поверхности объекта, свет точечного источника отражается от идеального рассеивателя по закону Ламберта: интенсивность отражённого света пропорциональна косинусу угла между направлением света и нормалью к поверхности (Рисунок 1.1)

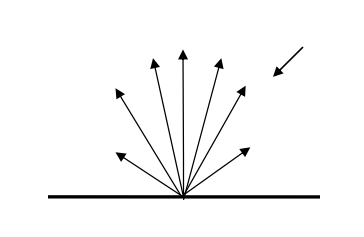


Рисунок 1.2 – Диффузионная модель отражения света

При диффузионной модели отражения света поверхность предметов выглядит блеклой, матовой. Части фигуры на который не падает свет кажутся чёрными, а на которые падает рассеивается, рассеянному свету соответствует распределенный источник. Поскольку для расчёта таких источников света требуются большие вычислительные затраты, в машинной графике (наш случай) они заменяются на коэффициент рассеяния – константу (k). Пример диффузионной модели отражения света (Рисунок 1.2)

Не лишним будет сказать, что для отображения цвета нашей поверхности мы будет использовать цветовую модель RGB. За основу этой цветовой модели принято три основных цвета R(Red), G(Green), B(Blue). В модели RGB любой цвет получается в результате сложения этих трех основных цветов.

Для модели RGB каждая из компонент может представляться числами, с ограниченным диапазоном от [0; 255;] так называемым - True Color в котором каждая компонента представлена в виде байта, что даёт 256 градаций для каждой компоненты.

Количество возможных сочетаний цветов составляет 256\*256\*256 = 16.7 млн цветов ().

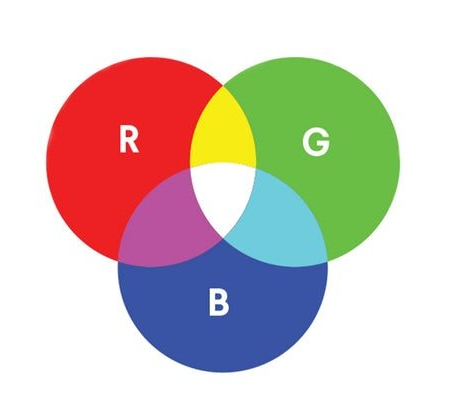


Рисунок 1.3 - Цветовая модель RGB

# ОБЩАЯ СХЕМА ПРИЛОЖЕНИЯ

Для решения поставленных задач было создано приложение MFC, построенное по следующей схеме:

Начало

Установка параметров в начальные значения, создание окна, запуск прослушивателей сообщений

Получение и обработка команд, прорисовка изображения с учетом поступающих параметров

Закрытие окна

Завершение обработки сообщений, закрытие окон, освобождение памяти

Конец

нет

да

Рисунок 2.1 – Блок-схема приложения

Создание окна и запуск прослушивателей сообщений выполняются по стандарту MFC приложений без использования архитектуры «Документ/Вид». Для хранения параметров была выработана соответствующая структура (приложение А). Завершение работы приложения выполняется также по стандарту MFC приложений.

Начало

Загрузка файла с координатами по умолчанию

Форматирование эллипсоида при помощи диалогового окна

Отображение эллипсоида с новыми параметрами

Конец

Команда меню “Выход”

Рисунок 2.2- Цикл получения и обработки команд

При помощи меню реализован вызов диалога загрузки данных с файла, а также настройка параметров для отображения эллипсоида в окне.

В диалоге указываются размер эллипсоида по координатам XYZ, выбор цвета, каким будет залит эллипсоид.

Сохранение получаемого изображения возможно вызовом соответствующего пункта меню. Происходит создание графического bmp-файла с соответствующими заголовками. Путь к файлу указывается при помощи файлового диалога. Последовательность действий по созданию bmp-файла представлена на рисунке 2.3.

|  |
| --- |
| Получение дескриптора окна и создание битовой карты по размеру рабочей области  Заполнение файлового и информационного заголовков  Запись в файл заголовков и аппаратно-независимой битовой карты |
| Рисунок 2.3- Описание алгоритма сохранения изображения |

# МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ АЛГОРИТМОВ

**АЛГОРИТМ ПОСТРОЕНИЯ ЭЛЛИПСОИДА С УЧЁТОМ КООРДИНАТ НАБЛЮДАТЕЛЯ**

|  |  |  |
| --- | --- | --- |
|  |  |  |

Для начала зададим произвольное значение полуосей эллипсоида a, b, c примем их значение за: .

Введем значения наблюдателя *PVIEW* и *PLIGHT*.

*, где 450 – ,0 – , 0 – ;*

Для отрисовки эллипсоида с учётом положения наблюдателя и источника света мы произведем преобразования *PVIEW* и *PLIGHT* в декартовы системы координат.

Так же нам нужно перевести *PVIEW* в видовую систему координат по

формуле:

MV =

По сформированному слою точек *Vertices* эллипсоида на промежутке от и мы проведем пересчет в оконную систему координат, с учётом рассчитанной позиции наблюдателя.

И пересчитаем их в оконную систему координат для отображения их в окне нашего приложения.

Так как в полигонах, где одна из вершин лежит на точке пересечения вершин с ординатой, отрисовываем по 3 вершины в полигоне, а у остальных 4, разделим наш дальнейший алгоритм на 3 части:

* обход верхней шапки;
* боковые полигоны;
* обход нижней шапки.

Приведем пример одной из частей обхода, так как они будут отличаться только кол-вом граней в полигонах.

Для начала проверяем видима ли наблюдателю наш полигон.

Для этого мы будем использовать такие параметры как:

*(Количество полигонов шапки)*

*(Количество полигонов шапки)*

После всех вычислений проверяем если SM >= 0, то грань видима, следовательно мы ее отрисовываем.

Если SM < 0 то, следовательно, не видима и мы ее не отрисовываем.

И уже на основе ранее полученных k, i отрисуем полигон, с учётом коэффициента освещенности.

, где Vecrticles сформированный слой точек

эллипса в оконной системе координат.

И соответственно по полученным координатам отрисовываем полигон на рабочей области.

**АЛГОРИТМ РАСЧЕТА КОЭФИЦИЕНТА ОСВЕЩЕННОСТИ**

*, где 450 – ,0 – , 0 – ;*

Для начала узнаем попадает ли на рассматриваемый полигон свет, для этого посчитаем *LSM.*

Если *LSM* >= 0, то на данный полигон падает наш свет, и далее в зависимости от того какой тип поверхности у нашей поверхности диффузионная или зеркальная будет высчитывать интенсивность падающего света.

Для расчета диффузионного отражения:

, где – интенсивность падающего света, - коэффициент диффузионного освещения;

Для расчёта зеркального отражения:

, с учётом пересчёта координат наблюдателя, где – коэффициент зеркального освещения, а степень скалярного произведения векторов задается в промежутке от [0;200], в зависимости от того какая отражающая способность нашей поверхности.

В нашем случае принимаем этот коэффициент за 20.

При получении коэффициентов, далее оперируем с ними по средством программной реализации. (Brush)

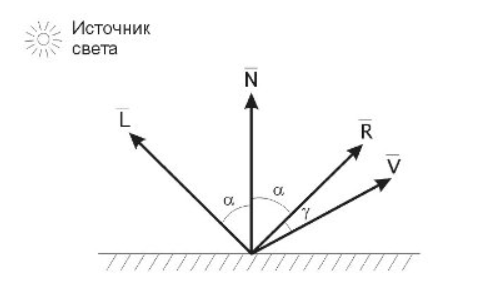


Рисунок 3.1 – Расчёт интенсивности отражённого света

# ПОЯСНЕНИЯ К ЛИСТИНГАМ ПРОГРАММЫ

**Листинг 1**

*void CChildView::OnSize(UINT nType, int cx, int cy) {*

*CWnd::OnSize(nType, cx, cy);*

*cWindow.SetRect(200, 200, cx - 100, cy - 100);*

*}*

В этом участке кода, происходит масштабирование эллипсоида с учётом изменения размеров окна. Метод *OnSize()* будет вызываться каждый раз, когда будет меняться размер рабочей области, и тем самым производить перерасчёт с учётом новых координат cx и cy, тем самым присваивая новые значения нашей рабочей области *(top,bottom,left,right)*.

**Листинг 2**

*void CChildView::OnKeyDown(UINT nChar,UINT nRepCount,UINT nFlags){*

*if(nChar==37){*

*PView(1)+=step;*

*if (PView(1)>360)*

*PView(1)-=360;*

*Invalidate();*

*}}*

На этом участке кода происходит поворот камеры наблюдателя влево, под номером *«37»* в char системе идет буква *A*, и при нажатии на любую клавишу будет вызываться этот метод, далее проверяться какая клавиша нажата если ее номер «37» то будет производиться увеличение нашего значения наблюдателя на «step», в нашем случае step = 5. Аналогичным образом будет производиться поворот камеры источника света, и изменение модели отражения.

**Листинг 3**

*CEllipsoid::CEllipsoid(int a,int b,int c)*

*{*

*Verticles.RedimMatrix(4,NoV); //точки в МСК*

*int A=a,B=b,C=c,i=1; //заданные параметры эллипсоида*

*Verticles(0,0)= 0; //самая верхняя точка*

*Verticles(1,0)= 0;*

*Verticles(2,0)= -C;*

*Verticles(3,0)= 1; //формирование эллипсов-слоев точек*

*for(double beta=-pi/2+pi/16;beta<=pi/2-3\*pi/32;beta+=pi/16){*

*//формирование эллипса-слоя на нужной высоте*

*for(double alpha=0;alpha<pi\*2-pi/32;alpha+=pi/16){*

*Verticles(0,i)=A\*cos(beta)\*cos(alpha); //x*

*Verticles(1,i)=B\*cos(beta)\*sin(alpha); //y*

*Verticles(2,i)=C\*sin(beta); //z*

*Verticles(3,i)=1;*

*i++;}}}*

Конструктор эллипсоида, с формированием эллипсов-слоев точек.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта мы реализовали Windows приложения, которое строит поверхность второго порядка с рядом изменяемых параметров таких как:

* цвет источника света;
* значение полуосей эллипсоида;
* координаты наблюдателя, и света;
* изменение размеров окна с масштабированием

При построении мы использовали основные математические алгоритмы построения поверхности второго порядка, пересчет интенсивности света: диффузионного и зеркального.

Реализовали эти алгоритмы программным способом используя при построении полигональную модель поверхности.

В конечном итоге имеем полностью функционирующее приложение, которое отвечает всем требования технического задания.

# ПРИЛОЖЕНИЕ 1

void CEllipsoid::DrawEnlighted(CDC& dc,CMatrix& PView,CMatrix& PLight,CRect& RW,COLORREF col,int m){

//цветоделение на RGB

double R=GetRValue(col),G=GetGValue(col),B=GetBValue(col);

//параметры освещенности

double Kd=1,Ks=1,I=1,Lights=1;

//перевод сферических координат наблюдателя в декартовы

CMatrix ViewCart=SphereToCart(PView);

//перевод сферических координат источнока света в декартовы

CMatrix LightCart=SphereToCart(PLight);

//матрица пересчета МСК->ВСК

CMatrix MV=CreateViewCoord(PView(0),PView(1),PView(2));

//матрица пересчета ВСК->ОСК

CMatrix K=SpaceToWindow(rs,RW);

//точки в ВСК

CMatrix ViewVert(4,NoV);

//точки в ОСК

CPoint MasVert[NoV];

for(int i=0;i<NoV;i++){

//пересчет координат точек МСК->ВСК

CMatrix V=MV\*Verticles.GetCol(i);

V(2)=1;

//пересчет координат точек ВСК->ОСК

CMatrix W=K\*V.GetCol(0,0,2);

MasVert[i].x=W(0);

MasVert[i].y=W(1);

}

CMatrix VE(3),R1(3),R2(3),V1(3),V2(3),VN(3);

double sm;

//обход верхней "шапки"

for(int i=0;i<32;i++){

int k=i+1;

if(k>=32)k=0;

R1=Verticles.GetCol(i+1,0,2); //текущая точка

R2=Verticles.GetCol(k+1,0,2); //следующая точка

VE=Verticles.GetCol(0,0,2); //самая верхняя точка

V1=R2-R1;V2=VE-R1; //ребра грани

VN=VectorMult(V2,V1); //вектор нормали

sm=ScalarMult(VN,ViewCart);

if(sm>=0){ //определение видимости грани

double lsm = ScalarMult(VN, LightCart);

if(lsm>=0) { //определение освещенности грани

//расчет освещенности для диффузионной модели

if(m==0)Lights=(double)I\*Kd\*cosViV2(VN,LightCart);

else if(m==1){

//расчет освещенности для зеркальной модели

double P=ScalarMult(VN,LightCart)/(VN.Abs()\*VN.Abs());

Lights=(double)I\*Ks\*pow(cosViV2(ViewCart,VN\*2.0\*P-LightCart),20);

}

} else Lights = 0;

CPoint\* p=new CPoint[3];

//получение точек грани

p[0]=MasVert[k+1];

p[1]=MasVert[i+1];

p[2]=MasVert[0];

//создание кисти расчитанной освещенности

CBrush\* br=new CBrush(RGB(R\*Lights,G\*Lights,B\*Lights));

dc.SelectObject(br);

//прорисовка полигона-грани

dc.Polygon(p,3);

//освобождение памяти

delete br;

}

}

//обход боковых граней

for(int j=1;j<14;j++){

for(int i=0;i<32;i++){

int k=i+1;

if(k>=32)k=0;

R1=Verticles.GetCol(i+j\*32+1,0,2); //текущая точка

R2=Verticles.GetCol(k+j\*32+1,0,2); //следующая точка

VE=Verticles.GetCol(i+(j-1)\*32+1,0,2); //точка предыдущего круга

V1=R2-R1;V2=VE-R1; //ребра грани

VN=VectorMult(V2,V1); //вектор нормали

sm=ScalarMult(VN,ViewCart);

if(sm>=0){ //определение видимости грани

double lsm = ScalarMult(VN, LightCart);

if(lsm>=0) { //определение освещенности грани

//расчет освещенности для диффузионной модели

if(m==0)Lights=(double)I\*Kd\*cosViV2(VN,LightCart);

else if(m==1){

//расчет освещенности для зеркальной модели

double P=ScalarMult(VN,LightCart)/(VN.Abs()\*VN.Abs());

Lights=(double)I\*Ks\*pow(cosViV2(ViewCart,VN\*2.0\*P-LightCart),20);

}

} else Lights = 0;

CPoint\* p=new CPoint[4];

//получение точек грани

p[0]=MasVert[k+j\*32+1];

p[1]=MasVert[i+j\*32+1];

p[2]=MasVert[i+(j-1)\*32+1];

p[3]=MasVert[k+(j-1)\*32+1];

//создание кисти расчитанной освещенности

CBrush\* br=new CBrush(RGB(R\*Lights,G\*Lights,B\*Lights));

dc.SelectObject(br);

//прорисовка полигона-грани

dc.Polygon(p,4);

//освобождение памяти

delete br;

}

}

//обход нижней "шапки"

for(int i=0;i<32;i++){

int k=i+1;

if(k>=32)k=0;

R1=Verticles.GetCol(i+13\*32+1,0,2); //текущая точка

R2=Verticles.GetCol(k+13\*32+1,0,2); //следующая точка

VE=Verticles.GetCol(449,0,2); //самая нижняя точка

V1=R2-R1;V2=VE-R1;

VN=VectorMult(V1,V2); //вектор нормали

sm=ScalarMult(VN,ViewCart);

if(sm>=0){ //определение видимости грани

double lsm = ScalarMult(VN, LightCart);

if(lsm>=0) { //определение освещенности грани

//расчет освещенности для диффузионной модели

if(m==0)Lights=(double)I\*Kd\*cosViV2(VN,LightCart);

else if(m==1){

//расчет освещенности для зеркальной модели

double P=ScalarMult(VN,LightCart)/(VN.Abs()\*VN.Abs());

Lights=(double)I\*Ks\*pow(cosViV2(ViewCart,VN\*2.0\*P-LightCart),20);

}

} else Lights = 0;

CPoint\* p=new CPoint[3];

//получение точек грани

p[0]=MasVert[k+13\*32+1];

p[1]=MasVert[i+13\*32+1];

p[2]=MasVert[449];

//создание кисти расчитанной освещенности

CBrush\* br=new CBrush(RGB(R\*Lights,G\*Lights,B\*Lights));

dc.SelectObject(br);

//прорисовка полигона-грани

dc.Polygon(p,3);

//освобождение памяти

delete br;

}

}

}}

# ПРИЛОЖЕНИЕ 2

void CChildView::OnPaint()

{

CPaintDC dc(this);

//прорисовка эллипсоида

pp->DrawEnlighted(dc,PView,PLight,c,col,m);

//прорисовки рамки выделения

if(m\_IsSelectedRect )

{

dc.SelectObject(pen);

dc.MoveTo(m\_SelectedRect->left,m\_SelectedRect->top);

dc.LineTo(m\_SelectedRect->right,m\_SelectedRect->top);

dc.LineTo(m\_SelectedRect->right,m\_SelectedRect->bottom);

dc.LineTo(m\_SelectedRect->left,m\_SelectedRect->bottom);

dc.LineTo(m\_SelectedRect->left,m\_SelectedRect->top);

}

}

# ПРИЛОЖЕНИЕ 3

void CChildView::OnSave1()

{

CFileDialog fOpenDlg(FALSE, \_T("fif"), 0, NULL, \_T("Image File (\*.bmp)|\*.bmp|"), this);

// заголовок диалога

fOpenDlg.m\_pOFN->lpstrTitle=\_T("Select BMP File");

// отображение диалога

if(fOpenDlg.DoModal()==IDOK)

{

m\_saved\_file = fOpenDlg.GetPathName();

Invalidate();

Sleep(1000);

ClientToBmp(GetSafeHwnd(), m\_saved\_file, \*m\_SelectedRect );

}

m\_IsSelectedRect = false;

}

# СПИСОК ЛИТЕРАТУРЫ

1. Секунов Н.Ю. Самоучитель Visual C++ .NET. – СПб.: БХВ-Петербург, 2002. – 736 с.: ил.
2. Давыдов В.Г. Visual C++. Разработка Windows-приложений с помощью MFC и API-функций. – СПб.: БХВ-Петербург, 2008. – 576 с.: ил.
3. Поляков А.Ю., Брусенцев В.А. Методы и алгоритмы компьютерной графики в примерах на Visual C++, 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2003. – 560 с.: ил.
4. Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002. – 432с.: ил.