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 «Тор»

Исполнитель

студент 3 курса 8 группы Савин В. А.

Руководитель

доцент, к. т.н. Дятко А. А.

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_ А. А. Дятко

|  |
| --- |
| Учреждение образования |
| «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ» |
| Факультет информационных технологий |
| Кафедра информатики и веб - дизайна |
| Специальность: 1-98 01 03 – «Программное обеспечение информационной безопасности мобильных систем» |
|  |
| Утверждаю |
| Заведующий кафедрой |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.М. Романенко |
| «\_\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 |

**ЗАДАНИЕ**

**на курсовой проект**

|  |  |
| --- | --- |
| **студенту** Савину В.А |  |
| **1. Тема** Тор | |

**2. Сроки защиты: до 15.12.2021**

**3. Исходные данные**

Лист с исходными данными прилагается

**4. Содержание пояснительной записки курсового проекта**

* Титульный лист
* Задание на курсовой проект
* Реферат
* Содержание
* Введение
* Основные разделы
* Заключение
* Список использованных источников
* Приложение (Листинги программ и CD с проектом на С++)

**5. Перечень графического, иллюстрационного материала**

* Блок – схемы алгоритмов.
* Скриншоты необходимых окон.

**6. Календарный график работы**

* Изучение задачи, исходных данных и разработка алгоритмов: 15.10.2021
* Программирование алгоритмов и оформление пояснительной записки: 30.11.2021
* Защита курсового проекта: 1.12.2021 – 15.12.2021

**7. Дата выдачи задания** 23.09.2021

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *А.А. Дятко.*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Оглавление**

[Задание 4](#_Toc89893739)

[Введение 5](#_Toc89893740)

[Математическое описание алгоритмов 7](#_Toc89893741)

[Пояснения к листингам программы 10](#_Toc89893742)

[Заключение 12](#_Toc89893743)

[Приложения 13](#_Toc89893744)

[Приложение А. Структура ball\_params. 13](#_Toc89893745)

[Приложение Б. Класс диалога настройки размера. 14](#_Toc89893746)

[Приложение В. Описание класса CMatrix. 16](#_Toc89893747)

[Приложение Г. Методы класса Тор. 16](#_Toc89893748)

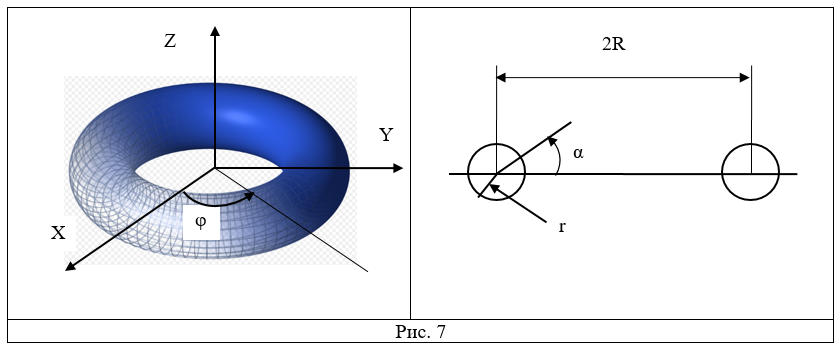
[Приложение Д. Функции аффинных преобразований. 19](#_Toc89893749)

[Приложение Е. Вспомогательные функции. 19](#_Toc89893750)

[Список литературы 21](#_Toc89893751)

# Задание

Построить 3D – изображение тора



Параметрическое уравнение тора в системе координат (СК) XYZ (Рис. 7)

|  |  |
| --- | --- |
|  | , |

Для описания поверхности тора использовать полигональную модель.

Изменяемые параметры модели:

* значения радиусов ;
* положение наблюдателя , в сферической СК с началом в точке O;
* положение источника света , в сферической СК с началом в точке O;
* цвет источника света;
* модель отражения света от поверхности фигуры.

Для установки параметров модели использовать окно диалога.

При изменении размера окна изображение соответствующим образом масштабируется.

Обеспечить запись изображения в графический файл \*.bmp

# Введение

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

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

Тор (тороид) — поверхность вращения, получаемая вращением образующей окружности вокруг оси, лежащей в плоскости этой окружности и не пересекающей её.

Обобщенно, тор — [топологическое пространство](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE) или [гладкое многообразие](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B4%D0%BA%D0%BE%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%B8%D0%B5), эквивалентное такой поверхности.

Иногда не требуют, чтобы ось вращения не пересекала образующую окружность. В таком случае, если ось вращения пересекает образующую окружность (или касается её), то тор называют закрытым, иначе открытым.

Уравнение тора с расстоянием от центра образующей окружности до оси вращения *R* и с радиусом образующей окружности *r* может быть задано параметрически в виде:

|  |  |
| --- | --- |
|  | , |

Также, в нашем приложении тор имеет диффузную модель отражения света от поверхности.

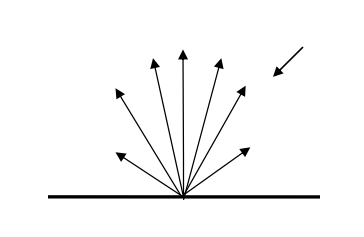


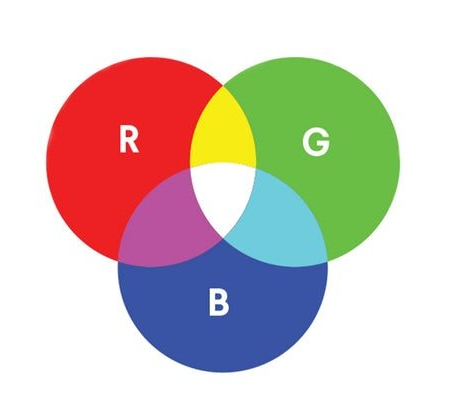
Рисунок 1 – Диффузионная модель отражения света

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)

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. (Рисунок 2)

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 2 - Цветовая модель RGB

# Математическое описание алгоритмов

**Алгоритм построения тора с учётом координат наблюдателя**

Для начала нам понадобится функции параметрического описания поверхности тора.

|  |  |
| --- | --- |
|  | , |

где R и r – большой и малый радиусы, и - широта и долгота. Для замкнутой поверхности углы и должны изменяться в полном круговом диапазоне, например, от 0 до 360 или от -180 до +180.

Для начала зададим произвольное значение малого и большого радиуса R, r примем их значение за: .

Введем значения наблюдателя PVIEW и PLIGHT.

, где 450 – ,0 – , 0 – ;

Для отрисовки тора 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 тора 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 количеством 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 тора в оконной системе координат.

И соответственно по полученным координатам отрисовываем полигон на рабочей области.

**Алгоритм расчёта коэффициента 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 высчитываем 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.

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

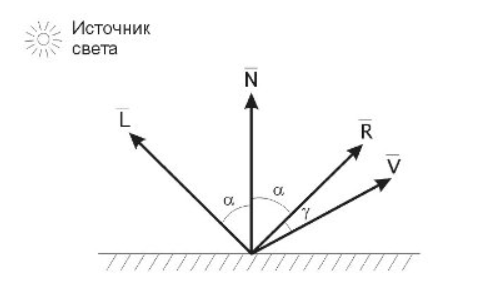


Рисунок 3 – Расчёт интенсивности отражённого света

# Описание алгоритмов решения задачи. 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

Поступила команда «Выход»

Завершение обработки сообщений, закрытие окон, освобождение памяти

Конец

нет

да

Создание окна и запуск прослушивателей сообщений выполняются по стандарту платформы MFC для приложений без использования архитектуры «Документ/Вид». Для хранения параметров, получаемых из диалога, для передачи их объекту класса тор модифицировался класс CChildView (приложение А). Завершение работы приложения выполняется также по стандарту платформы MFC.

## Получение и обработка команд

Цикл получения и обработки команд можно изобразить следующим образом:

Начало

Конец

Регуляция положения наблюдателя и источника света нажатием клавиш

Установка параметров в диалоге

(рисунок 2)

Перерисовка тора

Вызов справки из меню

Команда меню «Выход» или закрытие окна

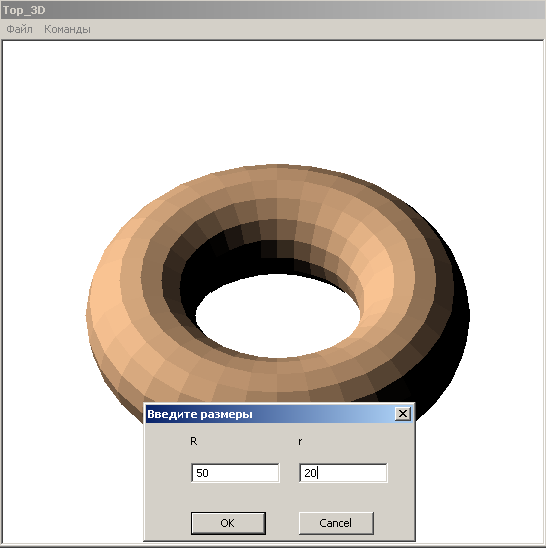
Отображение справочного окна

При помощи обработчика сообщений от клавиатуры реализованы следующие реакции:

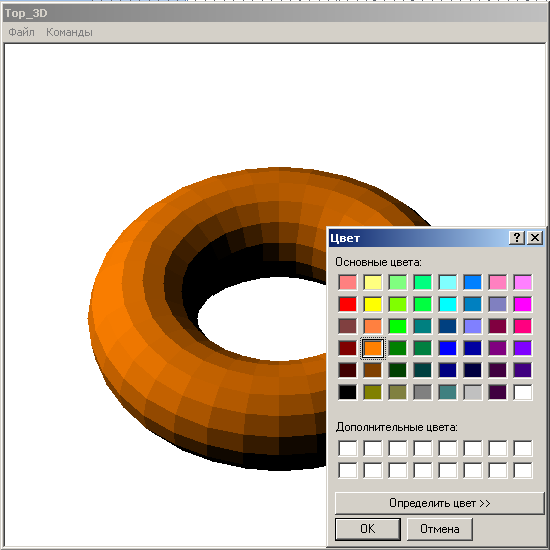
* поворот наблюдателя при помощи стрелок;
* поворот источника света при помощи клавиш w, a, s, d;
* вызов диалога настройки параметров при нажатии пробела (рисунок 2);
* выход из программы нажатием сочетания клавиш Alt+F4.

Кроме того, реакции вызова диалога настройки параметров, диалога справки, выхода доступны в меню.

В диалоге настройки параметров выполняется настройка всех требуемых параметров тора: размеров и цвета. Обработка действий отдельных элементов управления реализована при помощи механизма DoDataeXchange, являющегося полезным инструментом взаимодействия приложения на платформе MFC с пользователем (приложение Б).



а)



б)

Рисунок 4 – окно диалога настройки параметров

## Алгоритмы построения изображения

Одним из ключевых моментов решения поставленных задач является реализация координат точек в виде векторов в соответствии с математическим аппаратом компьютерной геометрии и графики. Для двухмерного пространства точка представляется в виде вектора из 3-ех элементов: двух координат и единицы; для трехмерного – трех координат и единицы. Это позволяет свести преобразования координат к умножению исходного вектора на соответствующую матрицу. Для работы с векторами и матрицами использовался класс CMatrix (приложение В).

Для решения задачи перемещения и поворота объектов в методах прорисовки поверхности тора с диффузионной моделью освещения были реализованы аффинные преобразования в соответствии с математическим аппаратом компьютерной геометрии и графики (приложение Г).

Для решения задачи изображения трехмерного процесса на двухмерном дисплее с учетом положения наблюдателя использовались матрицы пересчета из мировых координат в видовые и из видовых – в оконные. Расчет данных матриц выполнялся в соответствии с математическим аппаратом компьютерной геометрии и графики. Матрица пересчета из мировых координат в оконные формируется функцией CreateViewCoords, а матрица пересчета из видовых координат в оконные – функцией SpaceToWindow. Определения функций представлены в приложении Е.

Положение наблюдателя и источника света удобнее изменять в сферической системе координат при помощи стрелок и клавиш WASD. Но в расчетах в соответствии с математическим аппаратом компьютерной геометрии и графики требуется использовать координаты в декартовой системе. Поэтому был реализован метод перевода из сферических координат в декартовы математического аппарата компьютерной геометрии и графики. В диалогах регуляция положения наблюдателя и источника света не дублировалась.

Прорисовку поверхности тора по настроенным параметрам можно изобразить следующей блок-схемой:

Расчет матриц пересчета из мировой системы координат (МСК) в видовую и из видовой в оконную (ОСК)

Расположение точек тора в соответствии с параметрическими уравнениями и заданными радиусами

Порядка прорисовки полигонов тора «на наблюдателя»

Обход большого круга тора

Начало

Обход малых кругов тора

Вычисление положения источника света по заданным координатам в сферической системе координат, пересчет в оконные координаты, изображение источника света на экране

Конец

Построение четырехугольных полигонов

Прорисовка полигонов

# Пояснения к листингам программы

**Листинг 1**

*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 2**

*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;**}*

На этом участке кода создается диалог сохранения выделенной области в файл.

**Листинг 3**

*Тор::Тор(int R,int r){*

*Verticles.RedimMatrix(4,NoV);*

*int i=0;*

*//установка координат точек тора в зависимости от заданных радиусов*

*for(double beta=-pi;beta<=pi-pi/NoVb;beta+=pi/NoVb\*2){//обход большого круга*

*for(double alpha=0;alpha<pi\*2-pi/NoVm;alpha+=pi/NoVm\*2){//обход малых кругов*

*Verticles(0,i)=(R+r\*cos(alpha))\*cos(beta);*

*Verticles(1,i)=(R+r\*cos(alpha))\*sin(beta);*

*Verticles(2,i)=r\*sin(alpha);*

*Verticles(3,i)=1;*

*i++;*

*}*

*}*

*rs=CRectD(-100,-100,100,100);*

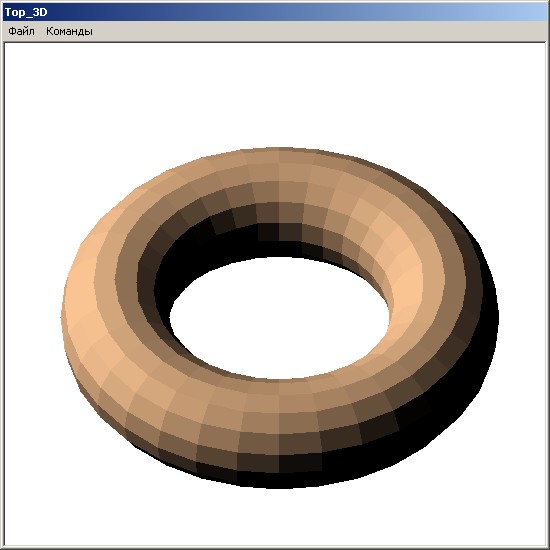
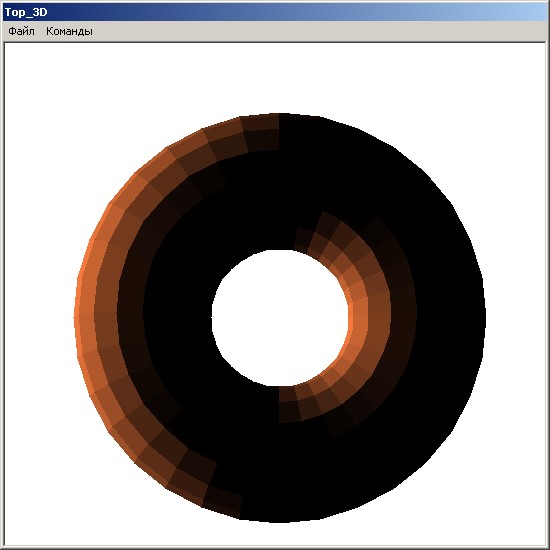
*}*

На этом участке кода создается конструктор тора с установкой координат точек тора в зависимости от заданных радиусов.

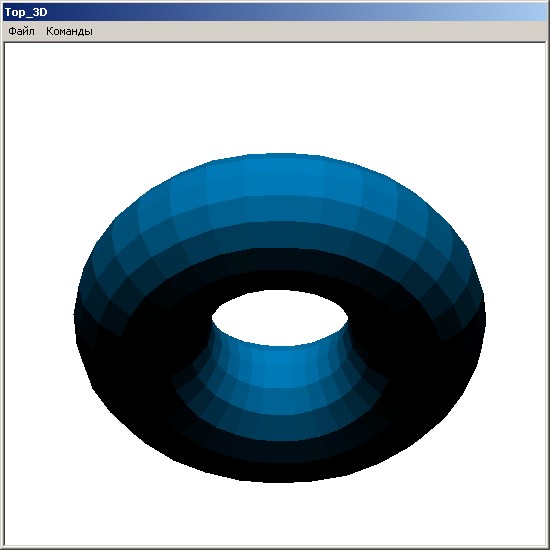
# Заключение

Разработанное приложение позволяет рассматривать трехмерный тор и следить за его освещением с различных сторон. В дальнейшем возможно усовершенствование приложения путем использования аппаратных средств пересчета координат и алгоритмов из оптимизированных библиотек. Получены навыки практического использования пакета MFC, применения математического аппарата компьютерной геометрии и графики.

Демонстрация работы приложения: движение тора в оранжевом освещении:

а) б)



в)

Рисунок 5 – демонстрация работы приложения.

# Приложения

Приложение А. Структура ball\_params.

*// ChildView.cpp : implementation of the CChildView class*

*//*

*#include "stdafx.h"*

*#include "Тор\_3D.h"*

*#include "ChildView.h"*

*#include "SizeDialog.h"*

*#ifdef \_DEBUG*

*#define new DEBUG\_NEW*

*#endif*

*// CChildView*

*CChildView::CChildView()*

*{*

*//начальные параметры тора, освещения, наблюдателя*

*step=10;*

*m=0;*

*PView.RedimMatrix(3);*

*PView(0)=450;PView(1)=0;PView(2)=45;*

*PLight.RedimMatrix(3);*

*PLight(0)=450;PLight(1)=90;PLight(2)=45;*

*col=RGB(255,200,150);*

*pp=new Тор();*

*}*

*CChildView::~CChildView()*

*{*

*}*

*BEGIN\_MESSAGE\_MAP(CChildView, CWnd)*

*ON\_WM\_PAINT()*

*ON\_WM\_KEYDOWN()*

*ON\_COMMAND(ID\_COMMANDS\_SETCOLOR, &CChildView::OnSetcolor)*

*ON\_COMMAND(ID\_COMMANDS\_SETSIZE, &CChildView::OnSetsize)*

*END\_MESSAGE\_MAP()*

*// CChildView message handlers*

*BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)*

*{*

*if (!CWnd::PreCreateWindow(cs))*

*return FALSE;*

*cs.dwExStyle |= WS\_EX\_CLIENTEDGE;*

*cs.style &= ~WS\_BORDER;*

*cs.lpszClass = AfxRegisterWndClass(CS\_HREDRAW|CS\_VREDRAW|CS\_DBLCLKS,*

*::LoadCursor(NULL, IDC\_ARROW), reinterpret\_cast<HBRUSH>(COLOR\_WINDOW+1), NULL);*

*return TRUE;*

*}*

*//перерисока тора*

*void CChildView::OnPaint()*

*{*

*CPaintDC dc(this); // device context for painting*

*pp->DrawEnlighted(dc,PView,PLight,CRect(0,0,550,550),col);*

*}*

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

*//движение наблюдателя на стрелки*

*if(nChar==37){ //влево*

*PView(1)+=step;*

*if(PView(1)>=360)PView(1)-=360;*

*Invalidate();*

*}*

*else if(nChar==39){ //вправо*

*PView(1)-=step;*

*if(PView(1)<0)PView(1)+=360;*

*Invalidate();*

*}*

*else if(nChar==38){ //вверх*

*PView(2)+=step;*

*if(PView(2)>=360)PView(2)-=360;*

*Invalidate();*

*}*

*else if(nChar==40){ //вниз*

*PView(2)-=step;*

*if(PView(2)<0)PView(2)+=360;*

*Invalidate();*

*}*

*//движение наблюдателя на WASD*

*else if(nChar==65){ //A*

*PLight(1)-=step;*

*if(PLight(1)>=360)PLight(1)-=360;*

*Invalidate();*

*}*

*else if(nChar==68){ //D*

*PLight(1)+=step;*

*if(PLight(1)<0)PLight(1)+=360;*

*Invalidate();*

*}*

*else if(nChar==87){ //W*

*PLight(2)-=step;*

*if(PLight(2)<0)PLight(2)+=360;*

*Invalidate();*

*}*

*else if(nChar==83){ //S*

*PLight(2)+=step;*

*if(PLight(2)>=360)PLight(2)-=360;*

*Invalidate();*

*}*

*}*

*//установка цвета через диалог*

*void CChildView::OnSetcolor()*

*{*

*CColorDialog ccd;*

*if(ccd.DoModal()==IDOK){*

*col=ccd.GetColor();*

*Invalidate();*

*}*

*}*

*//установка размера через диалог*

*void CChildView::OnSetsize()*

*{*

*SizeDialog sd;*

*if(sd.DoModal()==IDOK){*

*int R=sd.GetR(),r=sd.Getr();*

*delete pp;*

*pp=new Тор(R,r);*

*Invalidate();*

*}*

*}*

Приложение Б. Класс диалога настройки размера.

*// SizeDialog.cpp : implementation file*

*//*

*#include "stdafx.h"*

*#include "SizeDialog.h"*

*// SizeDialog dialog*

*IMPLEMENT\_DYNAMIC(SizeDialog, CDialog)*

*SizeDialog::SizeDialog(CWnd\* pParent /\*=NULL\*/)*

*: CDialog(SizeDialog::IDD, pParent)*

*{*

*EnableAutomation();*

*}*

*SizeDialog::~SizeDialog()*

*{*

*}*

*void SizeDialog::OnFinalRelease()*

*{*

*// When the last reference for an automation object is released*

*// OnFinalRelease is called. The base class will automatically*

*// deletes the object. Add additional cleanup required for your*

*// object before calling the base class.*

*CDialog::OnFinalRelease();*

*}*

*void SizeDialog::DoDataExchange(CDataExchange\* pDX)*

*{*

*CDialog::DoDataExchange(pDX);*

*DDX\_Control(pDX, IDC\_EDIT1, Redit);*

*DDX\_Control(pDX, IDC\_EDIT2, redit);*

*}*

*BEGIN\_MESSAGE\_MAP(SizeDialog, CDialog)*

*ON\_BN\_CLICKED(IDOK, &SizeDialog::OnBnClickedOk)*

*ON\_EN\_CHANGE(IDC\_EDIT1, &SizeDialog::OnEnChangeEdit1)*

*ON\_EN\_CHANGE(IDC\_EDIT2, &SizeDialog::OnEnChangeEdit2)*

*END\_MESSAGE\_MAP()*

*BEGIN\_DISPATCH\_MAP(SizeDialog, CDialog)*

*END\_DISPATCH\_MAP()*

*// Note: we add support for IID\_ISizeDialog to support typesafe binding*

*// from VBA. This IID must match the GUID that is attached to the*

*// dispinterface in the .IDL file.*

*// {140C0F47-65E4-4C40-9F5E-2B44453B3970}*

*static const IID IID\_ISizeDialog =*

*{ 0x140C0F47, 0x65E4, 0x4C40, { 0x9F, 0x5E, 0x2B, 0x44, 0x45, 0x3B, 0x39, 0x70 } };*

*BEGIN\_INTERFACE\_MAP(SizeDialog, CDialog)*

*INTERFACE\_PART(SizeDialog, IID\_ISizeDialog, Dispatch)*

*END\_INTERFACE\_MAP()*

*// SizeDialog message handlers*

*void SizeDialog::OnBnClickedOk()*

*{*

*// TODO: Add your control notification handler code here*

*OnOK();*

*}*

*void SizeDialog::OnEnChangeEdit1()*

*{*

*CString\* s=new CString();*

*Redit.GetWindowText(\*s);*

*R=atoi(s->GetBuffer());*

*}*

*void SizeDialog::OnEnChangeEdit2()*

*{*

*CString\* s=new CString();*

*redit.GetWindowText(\*s);*

*r=atoi(s->GetBuffer());*

*}*

*int SizeDialog::GetR(){*

*return R;*

*}*

*int SizeDialog::Getr(){*

*return r;*

*}*

Приложение В. Описание класса CMatrix.

*class CMatrix*

*{*

*double \*\*array; // Содержимое*

*int n\_rows; // Число строк*

*int n\_cols; // Число столбцов*

*public:*

*CMatrix(); // Конструктор по умолчанию (1 на 1)*

*CMatrix(int,int); // Конструктор*

*CMatrix(int); // Конструктор -вектора (один столбец)*

*CMatrix(const CMatrix&); // Конструктор копирования*

*~CMatrix();*

*double &operator()(int,int); // Выбор элемента матрицы по индексу*

*double &operator()(int); // Выбор элемента вектора по индексу*

*CMatrix operator-(); // Оператор "-"*

*CMatrix operator=(const CMatrix&); // Оператор "Присвоить": M1=M2*

*CMatrix operator\*(CMatrix&); // Оператор "Произведение": М1\*М2*

*CMatrix operator\*(double x); // Оператор "Произведение": М1\*a*

*CMatrix operator/(double x); // Оператор "Деление": М1/a*

*CMatrix operator+(CMatrix&); // Оператор "+": M1+M2*

*CMatrix operator-(CMatrix&); // Оператор "-": M1-M2*

*CMatrix operator+(double); // Оператор "+": M+a*

*CMatrix operator-(double); // Оператор "-": M-a*

*int rows()const{return n\_rows;} ; // Возвращает число строк*

*int cols()const{return n\_cols;}; // Возвращает число строк*

*CMatrix Transp(); // Возвращает матрицу,транспонированную к текущей*

*CMatrix GetRow(int); // Возвращает строку по номеру*

*CMatrix GetRow(int,int,int);*

*CMatrix GetCol(int); // Возвращает столбец по номеру*

*CMatrix GetCol(int,int,int);*

*CMatrix RedimMatrix(int,int); // Изменяет размер матрицы с уничтожением данных*

*CMatrix RedimData(int,int); // Изменяет размер матрицы с сохранением данных,*

*//которые можно сохранить*

*CMatrix RedimMatrix(int); // Изменяет размер матрицы с уничтожением данных*

*CMatrix RedimData(int); // Изменяет размер матрицы с сохранением данных,*

*//которые можно сохранить*

*double MaxElement(); // Максимальный элемент матрицы*

*double MinElement(); // Минимальный элемент матрицы*

*double Abs(); // Модуль вектора*

*};*

Приложение Г. Методы класса Тор.

*#include "StdAfx.h"*

*#include "Тор.h"*

*void Тор::DrawEnlighted(CDC& dc,CMatrix& PView,CMatrix& PLight,CRect& RW,COLORREF col){*

*try{*

*CPen\* pn=new CPen(PS\_NULL,0,RGB(0,0,0));*

*dc.SelectObject(pn);*

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

*double Kd=1,Ks=1,I=1,Lights=0;*

*//преобразование сферических координат наблюдателя в декартовы*

*CMatrix ViewCart=SphereToCart(PView);*

*//преобразование сферических координат источника света в декартовы*

*CMatrix LightCart=SphereToCart(PLight);*

*//матрица пересчета из мировых координат в видовые (МСК->ВСК)*

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

*//матрица пересчета из видовых в оконные(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++){*

*CMatrix V=MV\*Verticles.GetCol(i); //пересчет МСК->ВСК*

*V(2)=1;*

*CMatrix W=K\*V.GetCol(0,0,2); //пересчет ВСК->ОСК*

*double v0=V(0),v1=V(1),v2=V(2);*

*double w0=W(0),w1=W(1),w2=W(2);*

*for(int j=0;j<3;j++){*

*ViewVert(j,i)=W(j);*

*}*

*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;*

*int j1=(PView(1)/180\*NoVb/2);*

*if(j1>=NoVb)j1=0;*

*int j0=j1+NoVb/2;*

*if(j0>NoVb)j0-=NoVb;*

*if(PView(2)>180){*

*j0+=NoVb/2;*

*if(j0>=NoVb)j0-=NoVb;*

*j1+=NoVb/2;*

*if(j1>=NoVb)j1-=NoVb;*

*}*

*for(int jr=j0,jl=j0;;jr++,jl--){//обход большого круга*

*if(jr>=NoVb)*

*jr=0;*

*int lr=jr+1;//номер следующего малого круга*

*if(lr>=NoVb)*

*lr=0;*

*int i0=(PView(2)/180\*NoVm/2),i1=i0-1;*

*if(i1<0)i1=NoVm-1;*

*for(int i=i0;;i++){//обход малых кругов*

*if(i>=NoVm)i=0;*

*int k=i+1;//номер следующей точки малого круга*

*if(k>=NoVm)k=0;*

*R1=Verticles.GetCol(i+jr\*NoVm,0,2);//текущая точка на малом круге*

*R2=Verticles.GetCol(k+jr\*NoVm,0,2);//следующая точка на малом круге*

*VE=Verticles.GetCol(i+lr\*NoVm,0,2);//точка на следующем малом круге*

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

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

*sm=ScalarMult(VN,ViewCart);*

*if(sm>=0){//проверка видимости грани*

*if(ScalarMult(VN,LightCart)>=0){*

*//расчет освещенности для диффузионной модели освещения*

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

*}*

*else Lights=0;*

*//прорисовка полигона*

*CPoint\* p=new CPoint[4];*

*p[0]=MasVert[k+jr\*NoVm];*

*p[1]=MasVert[i+jr\*NoVm];*

*p[2]=MasVert[i+lr\*NoVm];*

*p[3]=MasVert[k+lr\*NoVm];*

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

*dc.SelectObject(br);*

*dc.Polygon(p,4);*

*delete br;*

*}*

*if(i==i1)break;*

*}*

*//*

*if(jl>=NoVb)jl=0;*

*if(jl<0)*

*jl=NoVb-1;*

*int ll=jl+1;//номер следующего малого круга*

*if(ll>=NoVb)*

*ll=0;*

*i0=(PView(2)/180\*NoVm/2);*

*i1=i0-1;*

*if(i1<0)i1=NoVm-1;*

*for(int i=i0;;i++){//обход малых кругов*

*if(i>=NoVm)i=0;*

*int k=i+1;//номер следующей точки малого круга*

*if(k>=NoVm)k=0;*

*R1=Verticles.GetCol(i+jl\*NoVm,0,2);//текущая точка на малом круге*

*R2=Verticles.GetCol(k+jl\*NoVm,0,2);//следующая точка на малом круге*

*VE=Verticles.GetCol(i+ll\*NoVm,0,2);//точка на следующем малом круге*

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

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

*sm=ScalarMult(VN,ViewCart);*

*if(sm>=0){//проверка видимости грани*

*if(ScalarMult(VN,LightCart)>=0){*

*//расчет освещенности для диффузионной модели освещения*

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

*//расчет освещенности для зеркальной модели освещения*

*}*

*else Lights=0;*

*//прорисовка полигона*

*CPoint\* p=new CPoint[4];*

*p[0]=MasVert[k+jl\*NoVm];*

*p[1]=MasVert[i+jl\*NoVm];*

*p[2]=MasVert[i+ll\*NoVm];*

*p[3]=MasVert[k+ll\*NoVm];*

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

*dc.SelectObject(br);*

*dc.Polygon(p,4);*

*delete br;*

*}*

*if(i==i1)break;*

*}*

*if(jr==j1)break;*

*}*

*delete pn;*

*}*

*catch(...){*

*}*

*}*

*Тор::Тор(void)*

*{*

*Verticles.RedimMatrix(4,NoV);*

*int R=60,r=20,i=0;*

*//установка координат точек тора в зависимости от радиусов*

*for(double beta=-pi;beta<=pi-pi/NoVb;beta+=pi/NoVb\*2){//обход большого круга*

*for(double alpha=0;alpha<pi\*2-pi/NoVm;alpha+=pi/NoVm\*2){//обход малых кругов*

*Verticles(0,i)=(R+r\*cos(alpha))\*cos(beta);*

*Verticles(1,i)=(R+r\*cos(alpha))\*sin(beta);*

*Verticles(2,i)=r\*sin(alpha);*

*Verticles(3,i)=1;*

*i++;*

*}*

*}*

*rs=CRectD(-100,-100,100,100);*

*}*

*Тор::Тор(int R,int r)*

*{*

*Verticles.RedimMatrix(4,NoV);*

*int i=0;*

*//установка координат точек тора в зависимости от заданных радиусов*

*for(double beta=-pi;beta<=pi-pi/NoVb;beta+=pi/NoVb\*2){//обход большого круга*

*for(double alpha=0;alpha<pi\*2-pi/NoVm;alpha+=pi/NoVm\*2){//обход малых кругов*

*Verticles(0,i)=(R+r\*cos(alpha))\*cos(beta);*

*Verticles(1,i)=(R+r\*cos(alpha))\*sin(beta);*

*Verticles(2,i)=r\*sin(alpha);*

*Verticles(3,i)=1;*

*i++;*

*}*

*}*

*rs=CRectD(-100,-100,100,100);*

*}*

*Тор::~Тор(void)*

*{*

*}*

Приложение Д. Функции аффинных преобразований.

*CMatrix Translate2D(double x,double y){ //смещение в двухмерной системе координат*

*CMatrix m(3,3);*

*m(0,0)=1;m(1,1)=1;m(2,2)=1;*

*m(0,2)=x;*

*m(1,2)=y;*

*return m;*

*}*

*CMatrix Rotate2D(double fi){ //поворот в двухмерной системе координат*

*CMatrix m(3,3);*

*double fi\_r=fi\*pi/180.0;*

*m(0,0)=cos(fi\_r);*

*m(1,1)=cos(fi\_r);*

*m(2,2)=1;*

*m(0,1)=-sin(fi\_r);*

*m(1,0)=sin(fi\_r);*

*return m;*

*}*

*//смещение в трехмерной системе координат*

*CMatrix Translate3D(double x,double y,double z){*

*CMatrix m(4,4);*

*m(0,0)=1;m(1,1)=1;m(2,2)=1;m(3,3)=1;*

*m(0,3)=x;*

*m(1,3)=y;*

*m(2,3)=z;*

*return m;*

*}*

*//поворот вокруг оси ОZ*

*CMatrix Rotate3Dz(double fi){*

*CMatrix m(4,4);*

*double fi\_r=fi\*pi/180.0;*

*m(0,0)=cos(fi\_r);*

*m(1,1)=cos(fi\_r);*

*m(2,2)=1;m(3,3)=1;*

*m(0,1)=-sin(fi\_r);*

*m(1,0)=sin(fi\_r);*

*return m;*

*}*

*//поворот вокруг оси ОХ*

*CMatrix Rotate3Dx(double fi){*

*CMatrix m(4,4);*

*double fi\_r=fi\*pi/180.0;*

*m(1,1)=cos(fi\_r);*

*m(2,2)=cos(fi\_r);*

*m(0,0)=1;m(3,3)=1;*

*m(1,2)=-sin(fi\_r);*

*m(2,1)=sin(fi\_r);*

*return m;*

*}*

*//поворот вокруг оси ОY*

*CMatrix Rotate3Dy(double fi){*

*CMatrix m(4,4);*

*double fi\_r=fi\*pi/180.0;*

*m(0,0)=cos(fi\_r);*

*m(2,2)=cos(fi\_r);*

*m(1,1)=1;m(3,3)=1;*

*m(0,2)=-sin(fi\_r);*

*m(2,0)=sin(fi\_r);*

*return m;*

*}*

Приложение Е. Вспомогательные функции.

*#pragma once*

*#include "stdafx.h"*

*#include "MyGDI.h"*

*//получение оконных координат по видовым*

*CMatrix SpaceToWindow(CRectD& rs, CRect& rw){*

*CMatrix m(3,3);*

*double kx=(rw.right-rw.left)/(rs.right-rs.left);*

*double ky=(rw.bottom-rw.top)/(rs.top-rs.bottom);*

*m(2,2)=1;*

*m(0,0)=kx;*

*m(1,1)=-ky;*

*m(0,2)=rw.left-kx\*rs.left;*

*m(1,2)=rw.top-ky\*rs.bottom;*

*return m;*

*}*

*//векторное произведение векторов*

*CMatrix VectorMult(CMatrix V1,CMatrix V2){*

*CMatrix m(3),a(3,3);*

*a(0,1)=-V1(2);*

*a(0,2)=V1(1);*

*a(1,2)=-V1(0);*

*a(1,0)=V1(2);*

*a(2,0)=-V1(1);*

*a(2,1)=V1(0);*

*m=a\*V2;*

*return m;*

*}*

*//скалярное произведение векторов*

*double ScalarMult(CMatrix V1,CMatrix V2){*

*return V1(0)\*V2(0)+V1(1)\*V2(1)+V1(2)\*V2(2);*

*}*

*//косинус угла между векторами*

*double cosViV2(CMatrix V1,CMatrix V2){*

*return ScalarMult(V1,V2)/(V1.Abs()\*V2.Abs());*

*}*

*//получение декартовых координат по сферическим*

*CMatrix SphereToCart(CMatrix DView){*

*CMatrix Ve(3);*

*double fi\_r=(DView(1))\*pi/180;*

*double theta\_r=DView(2)\*pi/180;*

*Ve(0)=DView(0)\*sin(theta\_r)\*cos(fi\_r);*

*Ve(1)=DView(0)\*sin(theta\_r)\*sin(fi\_r);*

*Ve(2)=DView(0)\*cos(theta\_r);*

*return Ve;*

*}*

*//получение видовых координат по положению наблюдателя*

*CMatrix CreateViewCoord(double R,double fi,double theta){*

*CMatrix Ve(4,4),Mz(4,4);*

*Mz(0,0)=-1;Mz(1,1)=1;Mz(2,2)=1;Mz(3,3)=1;*

*double fi\_r=fi\*pi/180;*

*double theta\_r=theta\*pi/180;*

*Ve=Mz\*Rotate3Dx(180-theta)\*Rotate3Dz(90-fi);*

*return Ve;*

*}*

# Список литературы

1. Поляков А., Брусенцев В. - Методы и алгоритмы компьютерной графики в примерах на Visual C++ - СПб.: БХВ-Петербург, 2003.
2. Давыдов В. - Visual C++. Разработка Windows-приложений с помощью MFC и API-функций.
3. Шикин Е.В., Боресков А.В. - Компьютерная графика. Динамика, реалистические изображения. - М.: Диалог-МИФИ, 1995.
4. Павлидис Т. - Алгоритмы машинной графики и обработка изображений. - М.: Радио и связь, 1988.