ГОСУДАРСТВЕННАЯ СЛУЖБА ГРАЖДАНСКОЙ АВИАЦИИ
МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей.

Курсовая работа защищена с оценкой_____

Подпись руководителя _____

Курсовая работа

По дисциплине «Компьютерная графика»

Выполнил: студент группы ЭВМ 5-2
Максимкин А. В.
Проверила: Федотова Т. Н.

Москва 2006 г.

Содержание

1. Техническое задание

Разработать программу геометрического моделирования процесса вращения трехмерного объекта вокруг центра масс данного объекта и произвольного вектора.

Объект – сфера, состоящая из 320 граней.

Вид проецирования – центральное одноточечное, параллельное ортографическое.

В программе предусмотреть удаление невидимых ребер объекта по алгоритму Робертса, а также его закраску.

Программа должна обладать удобным пользовательским интерфейсом, возможностью оперативного ввода исходных данных, а также подсказками по элементам управления объектами и их изображениями в пространстве.

2. Возможности программы

Данная программа позволяет увидеть математически построенную сферу из икосаэдра, произвести ее вращение по шести направлениям относительно центра масс, вращение вокруг произвольного вектора в двух направлениях (прямом и обратном). Также программа имеет возможность удаления невидимых линий сферы, производить ее закрашивание, цветом, выбранным пользователем из палитры цветов. Также пользователь может изменять цвет вектора и граней сферы. Программа содержит удобное меню, позволяющее пользователю произвести легкое и наглядное управление объектами. В результате запуска программы пользователь увидит на экране своего монитора сферу с 320 гранями при центральном одноточечном проецировании, а также слева экрана закладки, содержащие основные элементы управления объектами и подсказки по их использованию и назначению. В программе предусмотрены два вида проецирования – центральное одноточечное, а также параллельное ортографическое.

Программа была реализована в среде программирования Microsoft Visual Studio .NET 2003 с помощью системных функций API (Application Programming Interface – Интерфейс прикладного программирования).

3. Системные требования

Программа разрабатывалась на ЭВМ, имеющей следующие технические возможности:

Процессор: Intel Celeron CPU 1.80 GHz.
Оперативная память: 1GB.
Видеоадаптер: NVIDIA GeForce4 Ti 4200.
Монитор: Acer AL1716.
Переферийные устройства: Клавиатура, мышь

Среда разработки программы: Microsoft Visual C++.
Операционная система: Microsoft Windows XP.

Минимальные системные требования для корректного функционирования программы:

Мышь, процессор 500 MHz, оперативная память 256 Mb, монитор и видеоадаптер, способные поддерживать разрешение экрана 1280 на 1024 точек или большее (ВАЖНО!!! При меньшем разрешении программа будет выполняться не корректно.), операционная система Windows XP. Общий размер программы (включающий все ее файлы и модули) – 3, 30 Mb.

4. Математическая реализация задания

4.1 Изображение трехмерного объекта

Для изображения трехмерного объекта на экране необходимо выполнить два преобразования:

а) Видовое преобразование – эта операция обеспечивает переход от мировой системы координат с началом в центре объекта, к видовой системе координат, где начало системы координат совпадает с центром проецирования.

б) Перспективное преобразование – переход от видовых координат к экранным координатам.

Видовые координаты [xV[i], yV[i], zV[i]] определяющих точек трехмерного объекта при заданных сферических координатах точки наблюдения (tetta, phi, ro), которые пользователь может изменять в процессе выполнения программы, могут быть рассчитаны по заданным мировым координатам этих точек [xW[i], yW[i], zW[i]] с помощью выражения:

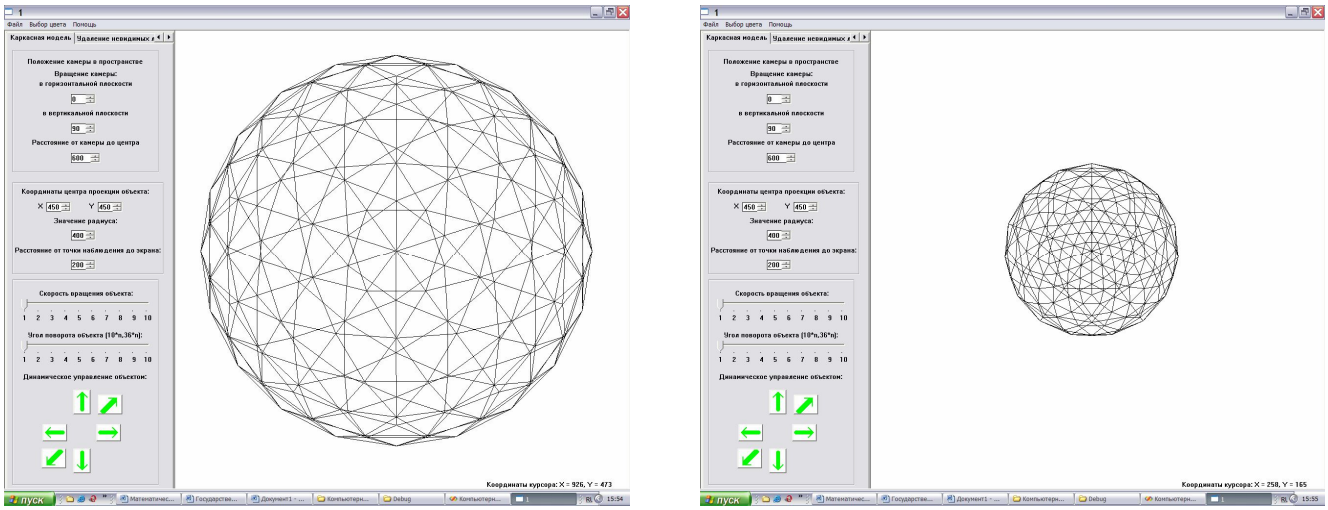
[xV yV zV 1] =[xW yW zW 1]*V, где V – мастрица видового преобразования (см. рисунок 1).

$$V = \begin{matrix} \sin(tetta) & \cos(phi)\cos(tetta) & \sin(phi)\cos(tetta) & 0 \\ \cos(tetta) & \cos(phi)\sin(tetta) & \sin(phi)\sin(tetta) & 0 \\ 0 & \sin(phi) & \cos(phi) & 0 \\ 0 & 0 & ro & 1 \end{matrix}$$

Рисунок 1. Матрица видового преобразования.

В данной курсовой работе используются два вида проецирования (рисунок 2):

- центральное одноточечное,
- параллельное ортографическое.



а) б)

Рисунок 2. а) – при параллельном ортографическом; б) – при центральном одноточечном.

Пользователь сам может выбирать в ходе выполнения программы, в каком виде проецирования будет изображаться искомый объект.

Центральное одноточечное проецирование выполняется с помощью математических выражений:

$$X=(xV/yV)*d+xx,$$
$$Y=(xV/yV)*d+yy, \text{ где}$$

(X, Y) – экранные координаты проецируемой точки;
(xV, yV) – видовые координаты точки;
d – расстояние от центра проецирования (точки наблюдения) до проекционной плоскости (экрана);
(xx, yy) – желаемые координаты центра проекции объекта на эктане.

Параллельное ортографическое проецирование позволяет показать объект в натуральную величену без каких либо видимых искажений в пространстве (как при центральном проецировании) и выполняется с помощью математических выражений:

$$X=xV+xx,$$
$$Y=yV+yy, \text{ где}$$

(X, Y) – экранные координаты проецируемой точки;
(xV, yV) – видовые координаты точки;
(xx, yy) – желаемые координаты центра проекции объекта на эктане.

Пользователь может изменять параметры xx, yy и d динамически в ходе выполнения программы.

4.2 Вращение сферы вокруг произвольного вектора

Вращение сферы вокруг произвольного вектора осуществляется с помощью результирующей матрицы.

Для подсчета данной результирующей матрицы необходимо перемножить матрицы элементарных геометрических преобразований в следующей последовательности:

1. Матрица переноса системы координат в начальную точку вектора:

$$T^{1} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ kxv & kvy & kvz & 1 \end{matrix}$$

$$[x^* \; y^* \; z^* \; 1]=[x \; y \; z \; 1]*T^{-1}$$

Здесь kvx, kvy, kvz – координаты начальной точки вектора.

2. Матрица поворота вокруг оси “z” на угол –tetta:

$$Rz \begin{matrix} \cos(\ tetta) & \sin(\ tetta) & 0 & 0 \\ \sin(\ tetta) & \cos(\ tetta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x** y** z** 1]=[x* y* z* 1]*Rz(-tetta)

3. Матрица поворота вокруг оси “y” на угол –phi:

$$Ry \begin{matrix} \cos(\ phi) & 0 & \sin(\ phi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\ phi) & 0 & \cos(\ phi) & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x*** y*** z*** 1]=[x** y** z** 1]*Ry(-phi)

4. Матрица поворота вокруг оси “z” на угол alfa:

$$Rz \begin{matrix} \cos(alfa) & \sin(alfa) & 0 & 0 \\ \sin(alfa) & \cos(alfa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x**** y**** z**** 1]=[x*** y*** z*** 1]*Rz(alfa)

5. Матрица поворота вокруг оси “y” на угол phi:

$$Ry \begin{matrix} \cos(phi) & 0 & \sin(phi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(phi) & 0 & \cos(phi) & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x***** y***** z***** 1]=[x**** y**** z**** 1]*Ry(phi)

6. Матрица поворота вокруг оси “z” на угол tetta:

$$Rz \begin{matrix} \cos(tetta) & \sin(tetta) & 0 & 0 \\ \sin(tetta) & \cos(tetta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x****** y****** z****** 1]=[x***** y***** z***** 1]*Ry(tetta)

7. Матрица переноса системы координат в прежнее положение:

$$T \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ kxv & kvy & kvz & 1 \end{matrix}$$

$$[x******* \; y******* \; z******* \; 1]=[x****** \; y****** \; z****** \; 1]*T^{1}$$

Здесь kvx, kvy, kvz – координаты начальной точки вектора.

В результате получается результирующая матрица G, имеющая следующий вид:

$$G=T^{-1}*Rz(-tetta)*Ry(-phi)*Rz(alfa)*Ry(phi)*Rz(tetta)*T^{1}$$

Для того, чтобы повернуть сферу вокруг произвольного вектора необходимо в результирующую матрицу подставить следующие выражения:

cos(-tetta)=cos(tetta); cos(-phi)=cos(phi);
sin(-tetta)=-sin(tetta); sin(-phi)=-sin(phi);

$$\cos(\text{tetta}) \quad \frac{kvx2 \quad kvx}{\sqrt{(kvx2 \quad kvx)\,(kvx2 \quad kvx) \quad (kvy2 \quad kvy)\,(kvy2 \quad kvy)}}$$

$$\sin(\text{tetta}) \quad \frac{(kvy2 \quad kvy)\,(kvy2 \quad kvy)}{\sqrt{(kvx2 \quad kvx)\,(kvx2 \quad kvx) \quad (kvy2 \quad kvy)\,(kvy2 \quad kvy)}}$$

$$\cos(\text{phi}) \quad \frac{(kvz2 \quad kvz)\,(kvz2 \quad kvz)}{\sqrt{(kvx2 \quad kvx)\,(kvx2 \quad kvx) \quad (kvy2 \quad kvy)\,(kvy2 \quad kvy)\,(kvz2 \quad kvz)}}$$

$$\sin(\text{phi}) \quad \frac{\sqrt{(kvx2 \quad kvx)\,(kvx2 \quad kvx) \quad (kvy2 \quad kvy)\,(kvy2 \quad kvy)}}{\sqrt{(kvx2 \quad kvx)\,(kvx2 \quad kvx) \quad (kvy2 \quad kvy)\,(kvy2 \quad kvy) \quad (kvz2 \quad kvz)\,(kvz2 \quad kvz)}}$$

Вычисление результирующей матрицы проводилось с помощью математического пакета программ Mathcad версии 11.0a. Далее приведены данные вычисления:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -kvx & -kvy & -kvz & 1 \end{pmatrix} \cdot \begin{pmatrix} ct & -st & 0 & 0 \\ st & ct & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cp & 0 & -sp & 0 \\ 0 & 1 & 0 & 0 \\ sp & 0 & cp & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(alfa) & \sin(alfa) & 0 & 0 \\ -\sin(alfa) & \cos(alfa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cp & 0 & sp & 0 \\ 0 & 1 & 0 & 0 \\ -sp & 0 & cp & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} ct & st & 0 & 0 \\ -st & ct & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ kvx & kvy & kvz & 1 \end{pmatrix} \rightarrow$$

$$\begin{bmatrix} [(ct\cdot cp\cdot\cos(alfa) + st\cdot\sin(alfa))\cdot cp + ct\cdot sp^{2}]\cdot ct - (ct\cdot cp\cdot\sin(alfa) - st\cdot\cos(alfa))\cdot st \\ [(st\cdot cp\cdot\cos(alfa) - ct\cdot\sin(alfa))\cdot cp + st\cdot sp^{2}]\cdot ct - (st\cdot cp\cdot\sin(alfa) + ct\cdot\cos(alfa))\cdot st \\ (sp\cdot\cos(alfa)\cdot cp - cp\cdot sp)\cdot ct - sp\cdot\sin(alfa)\cdot st \\ [[(-kvx\cdot ct - kvy\cdot st)\cdot cp - kvz\cdot sp]\cdot\cos(alfa) - (kvx\cdot st - kvy\cdot ct)\cdot\sin(alfa)]\cdot cp - [-(-kvx\cdot ct - kvy\cdot st)\cdot sp - kvz\cdot cp]\cdot sp]\cdot ct - [[(-kvx\cdot ct - kvy\cdot st)\cdot cp - kvz\cdot sp]\cdot\sin(alfa) + (kvx\cdot st - kvy\cdot ct)\cdot\cos(alfa)]\cdot st + kvx \end{bmatrix}$$

$$\begin{bmatrix} [(ct\cdot cp\cdot\cos(alfa) + st\cdot\sin(alfa))\cdot cp + ct\cdot sp^{2}]\cdot st + (ct\cdot cp\cdot\sin(alfa) - st\cdot\cos(alfa))\cdot ct \\ [(st\cdot cp\cdot\cos(alfa) - ct\cdot\sin(alfa))\cdot cp + st\cdot sp^{2}]\cdot st + (st\cdot cp\cdot\sin(alfa) + ct\cdot\cos(alfa))\cdot ct \\ (sp\cdot\cos(alfa)\cdot cp - cp\cdot sp)\cdot st + sp\cdot\sin(alfa)\cdot ct \\ [[(-kvx\cdot ct - kvy\cdot st)\cdot cp - kvz\cdot sp]\cdot\cos(alfa) - (kvx\cdot st - kvy\cdot ct)\cdot\sin(alfa)]\cdot cp - [-(-kvx\cdot ct - kvy\cdot st)\cdot sp - kvz\cdot cp]\cdot sp]\cdot st + [[(-kvx\cdot ct - kvy\cdot st)\cdot cp - kvz\cdot sp]\cdot\sin(alfa) + (kvx\cdot st - kvy\cdot ct)\cdot\cos(alfa)]\cdot ct + kvy \end{bmatrix}$$

$$\begin{bmatrix} (ct\cdot cp\cdot\cos(alfa) + st\cdot\sin(alfa))\cdot sp - ct\cdot sp\cdot cp & 0 \\ (st\cdot cp\cdot\cos(alfa) - ct\cdot\sin(alfa))\cdot sp - st\cdot sp\cdot cp & 0 \\ sp^{2}\cdot\cos(alfa) + cp^{2} & 0 \\ [[(-kvx\cdot ct - kvy\cdot st)\cdot cp - kvz\cdot sp]\cdot\cos(alfa) - (kvx\cdot st - kvy\cdot ct)\cdot\sin(alfa)]\cdot sp + [-(-kvx\cdot ct - kvy\cdot st)\cdot sp - kvz\cdot cp]\cdot cp + kvz & 1 \end{bmatrix}$$

Соответственно:
ct – cos(tetta);
st – sin(tetta);
cp – cos(phi);
sp – sin(phi).

4. 3 Вращение сферы вокруг осей “x”, “y”, “z” относительно центра масс

Для того, чтобы произвести вращение относительно центра масс объекта вокруг осей “x”, “y”, “z” необходимо воспользоваться матрицами трех элементарных геометрических преобразований, соответственно матрицы поворота вокруг осей “x”, “y” и “z”.

Матрицы поворота на угол alfa:

- вокруг оси “x”:

$$Rx = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & \cos(alfa) & \sin(alfa) & 0 \\ 0 & \sin(alfa) & \cos(alfa) & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x** y** z** 1]=[x* y* z* 1]*Rx

- вокруг оси “y”:

$$Ry = \begin{matrix} \cos(alfa) & 0 & \sin(alfa) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(alfa) & 0 & \cos(alfa) & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x** y** z** 1]=[x* y* z* 1]*Ry

- вокруг оси “z”:

$$Rz = \begin{matrix} \cos(alfa) & \sin(alfa) & 0 & 0 \\ \sin(alfa) & \cos(alfa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

[x** y** z** 1]=[x* y* z* 1]*Rz

Поворот осуществляется на угол alfa равный 1 или -1, в зависимости от того в какую сторону пользователь хочет повернуть сферу (рисунок 3). После поворота на угол в 1 или -1 градус координаты сферы запоминаются (старая сфера затирается) и на следующем шаге она выводится уже с новыми координатами – происходит эффект движения объекта в пространстве.

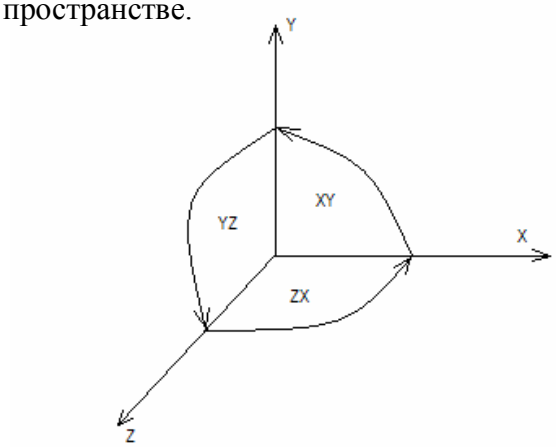


Рисунок 3. Положительные направления поворота.

4.4 Построение геометрической модели объекта

Следуя техническому заданию, в данной курсовой работе была построена геометрическая модель Платонова тела – объект сфера. Платоново тело – выпуклый многогранник, все грани которого суть правильные многоугольники и все многогранные углы при вершинах равны между собой.

Сфера может быть построена двумя способами:

1. Построение аналитической модели.
2. Использование икосаэдра для аппроксимации сферы.

В данной курсовой работе геометрическая модель сферы реализована с использованием второго способа.

Построение сферы

Данное построение удобно рассматривать как набор шагов, позволяющих достичь требуемого результата – построения геометрической модели сферы.

Шаг 1. Построение икосаэдра.

Рассечем круглый цилиндр радиуса a, ось которого совпадает с осью аппликат Z двумя плоскостями Z=-0.5*a и Z=0.5*a, где a – радиус будущего икосаэдра (рисунок 4).

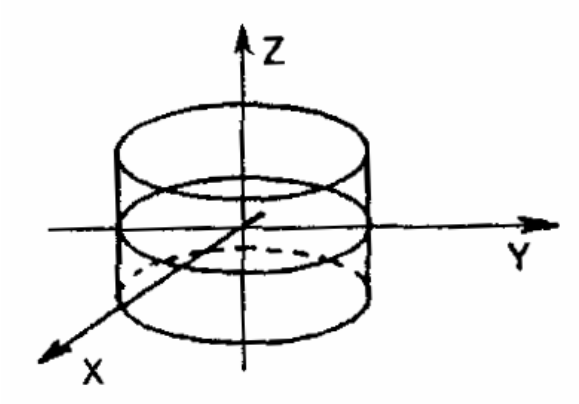


Рисунок 4. Рассеченный цилиндр.

Разобьем каждую из полученных окружностей на 5 равных частей так, как показано на рисунке 5.

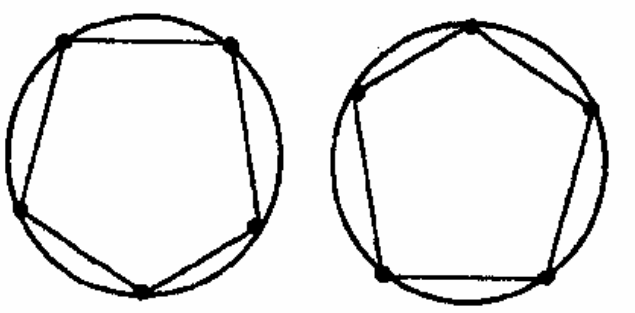


Рисунок 5. Деление окружностей на части.

Перемещаясь вдоль обеих окружностей против часовой стрелки, занумеруем выделенные 10 точек в порядке возрастания угла поворота (рисунок 6).

Рисунок 6. Нумерация точек на окружностях.

Затем последовательно, в соответствии с нумерацией, соединим эти точки прямолинейными отрезками (рисунок 7).

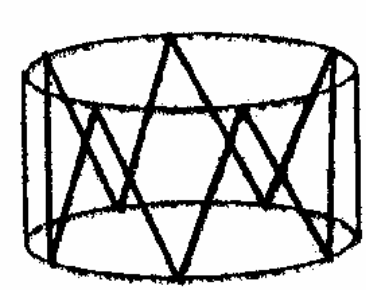


Рисунок 7. Результат соединения точек.

Стягивая теперь хордами точки, выделенные на каждой из окружностей, получаем в результате пояс из 10 правильных треугольников (рисунок 8).

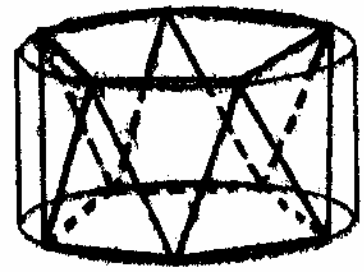


Рисунок 8. Результат соединения точек на окружностях.

Для завершения построения икосаэдра выберем на оси Z две точки так, чтобы длины боковых ребер пятиугольных пирамид с вершинами в этих точках и основаниями, совпадающими с построенными пятиугольниками (рисунок 9), были равны длинам сторон пояса.

Координаты Z этих точек $\pm \frac{\sqrt{5}}{2} \cdot a$

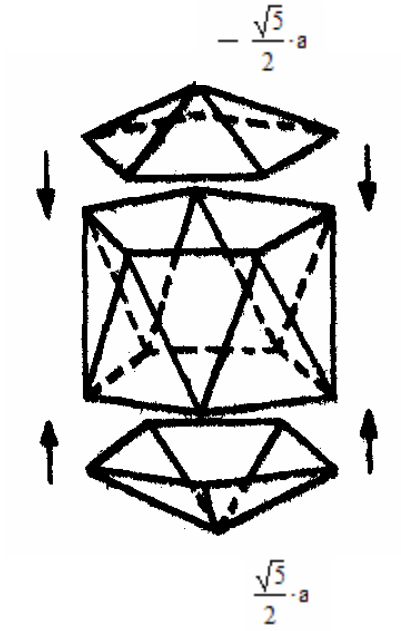


Рисунок 9. Верхняя и нижняя пятиугольные пирамиды с вершинами $-\frac{\sqrt{5}}{2}\cdot a$, $\frac{\sqrt{5}}{2}\cdot a$.

В результате описанных построений получаем 12 точек. Выпуклый многогранник с вершинами в этих точках будет иметь 20 граней, каждая из которых является правильным треугольником, и все его многогранные углы при вершинах будут равны между собой. Тем самым результат описанного построения – икосаэдр (рисунок 10). На рисунке 11 приведено программное построение икосаэдра.

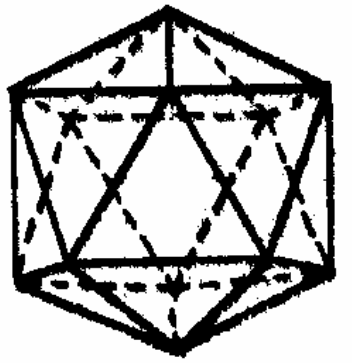


Рисунок 10. Икосаэдр.

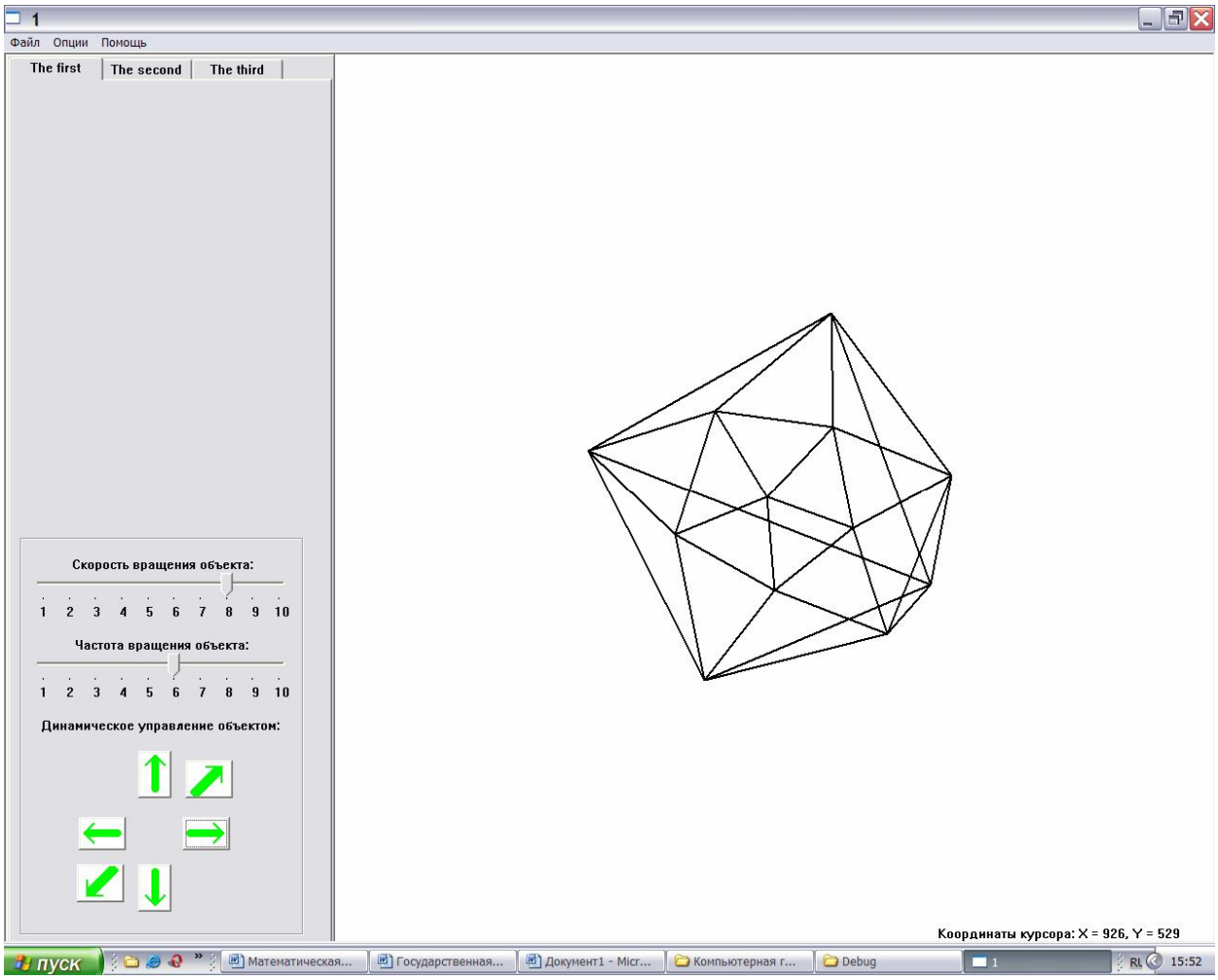


Рисунок 11. Построенный икосаэдр после поворотов вокруг осей X,Y и Z в программе.

Радиус икосаэдра – a.
Координаты точек, лежащих на окружностях рассеченного цилиндра вычисляются как:

X=a*cos(
Y=a*sin(.

Из них выбираются точки, условно лежащие на верхней и нижней окружности, в порядке чередования углов, кратных 36 градусам, начиная с нулевого угла.

Z=a/2 – для точек, лежащих на нижней окружности;
Z=-a/2 – для точек, лежащих на верхней окружности.

Координаты вершины верхней усеченной пирамиды:

X=0;
Y=0;
Z= $-\frac{\sqrt{5}}{2}\cdot a$.

Координаты вершины нижней усеченной пирамиды:

X=0;

Y=0;

Z= $\frac{\sqrt{5}}{2} \cdot a$.

Перед вычислением координат точек, лежащих на окружностях усеченного цилиндра необходимо преобразовать углы из градусов в радианы по формуле:

alfa(радианов)=alfa(градусов)*arctg(1)/45.0;

Шаг 2. Использование икосаэдра в качестве основы сферы, имеющей 80 треугольных граней.

Рассмотрим одну треугольную грань икосаэдра, представленную на рисунке 12.

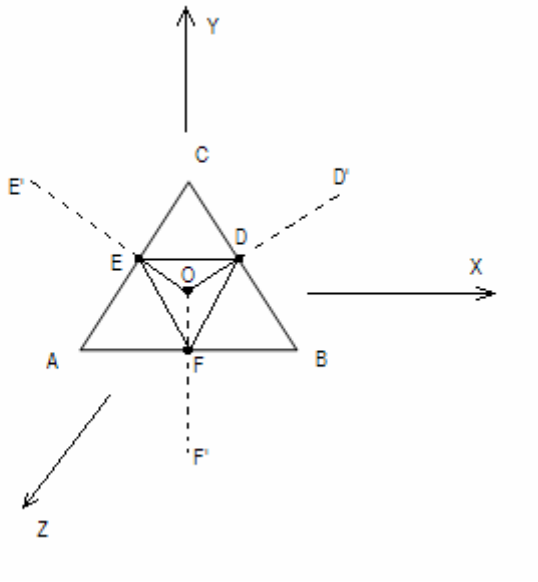


Рисунок 12. Треугольная грань икосаэдра.

Задача: Необходимо соединить координаты рассчитанных точек (E’, D’, F’) и вершин треугольной грани (A, B, C) как показано на рисунке 13.

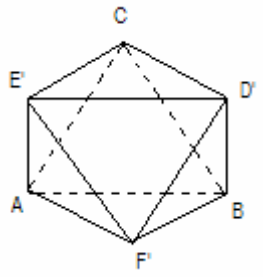


Рисунок 13. После аппроксимации.

1. Найдем середины ребер, образующих треугольную грань (A, B, C).
Рассмотрим общий случай подсчета координат середины отрезка (P1, P2), представленный на рисунке 14.

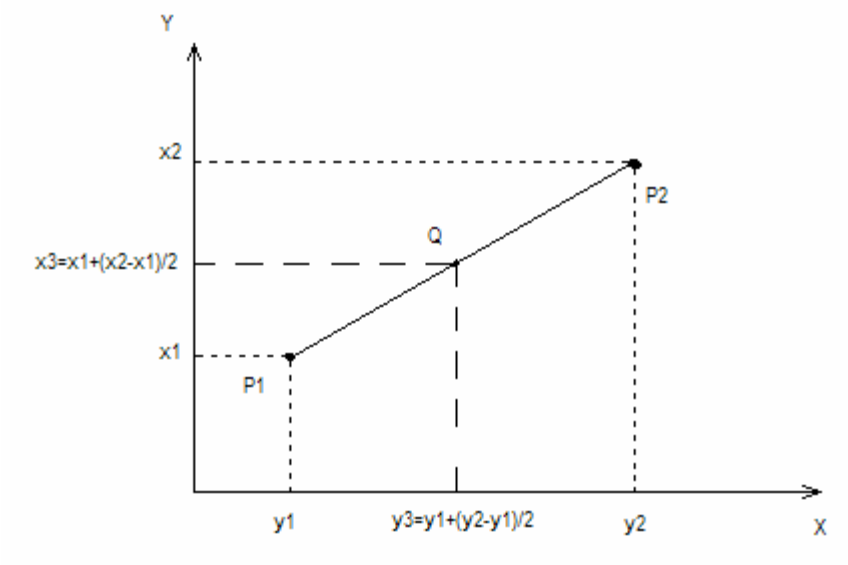


Рисунок 14. Определения середины отрезка.

Для того, чтобы найти координаты середины отрезка (P1,P2) необходимо к координатам точки P1 прибавить половину разности координат точек P2 и P1. Таким образом, будут получены необходимые координаты центра данного отрезка. Следовательно, координаты точки Q(x3,y3) будут равны:

$$x3=x1+ (x2-x1)/2;$$
$$y3=y1+ (y2-y1)/2.$$

Данным способом, используя координаты вершин треугольной грани (A, B, C) найдем координаты точек (D, E, F), лежащих на серединах отрезков (CB, AC, AB) соответственно.

2. Подсчитаем расстояние от центра икосаэдра до координат найденных точек (D, E, F). Данный расчет производится по формуле (на примере точки D):

$$d\,(OD)= \sqrt{X_D^2+Y_D^2+Z_D^2};$$

3. Повторим вычисления для оставшихся точек, составляющих грань – E и F.

4. Найдем координаты точек (D’,E’, F’) по формулам (на примере точки D’):

$$X_D'=X_D/d;\ Y_D'=Y_D/d;\ Z_D'=Z_D/d.$$

5. Повторим вычисления для оставшихся точек – E’ и F’.

6. Соединяем точки D’, E’, F’ по рассчитанным координатам как показано на рисунке 13.

Таким образом, аппроксимируем каждую грань икосаэдра, полученного на шаге 1. Данные вычисления необходимо провести с 20 гранями (60 точками).

Результат аппроксимации изображен на рисунке 15.

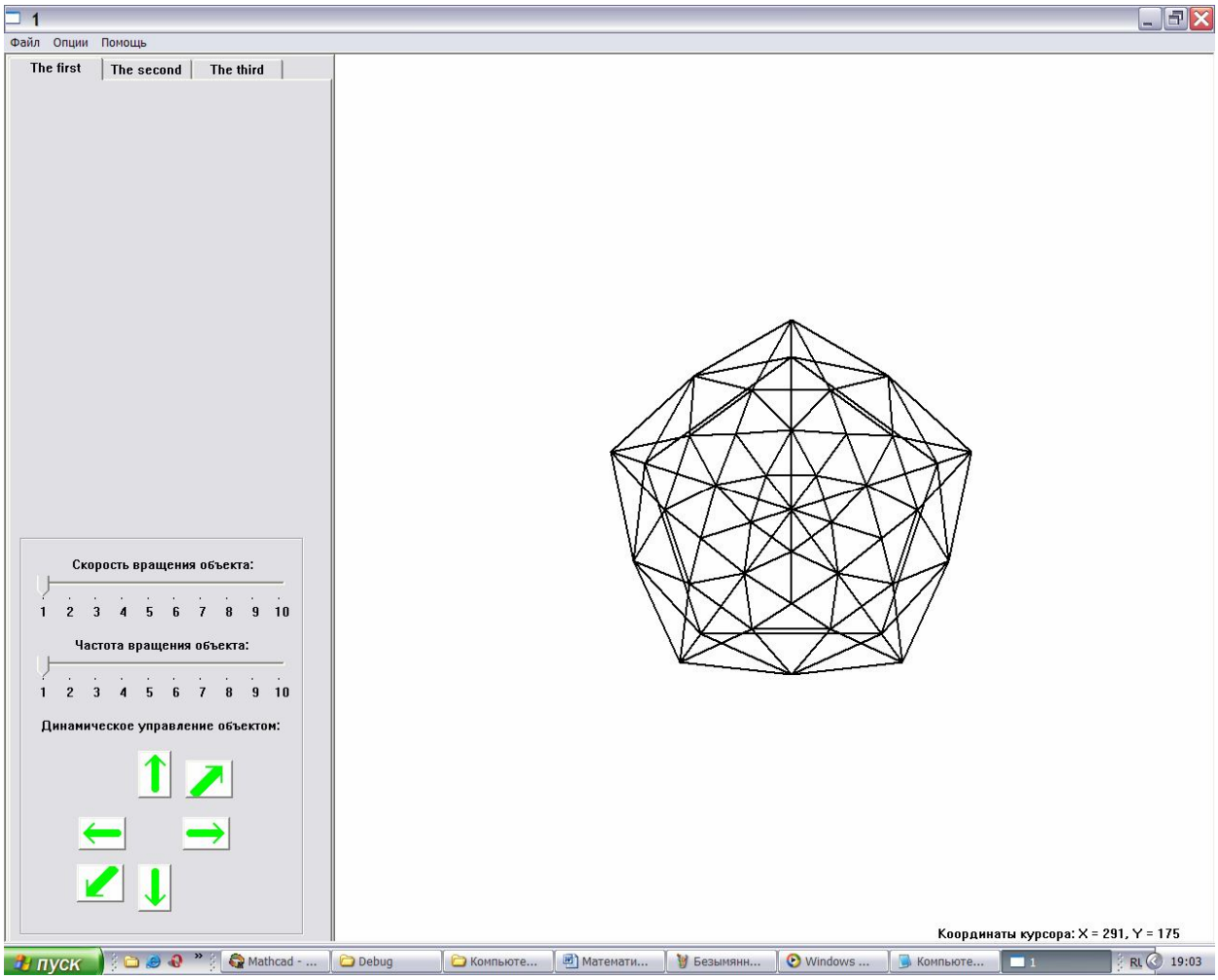


Рисунок 15. Построенная сфера с 80 гранями.

Шаг 3. . Использование сферы, полученной на шаге 2 в качестве основы сферы, имеющей 320 треугольных граней.

Повторить для каждой грани сферы, пункты 1-6 на шаге 2.
Данные вычисления необходимо провести с 80 гранями (240 точками).

В результате получим сферу с 320 гранями и 960 вершинами, представленную на рисунке 16.

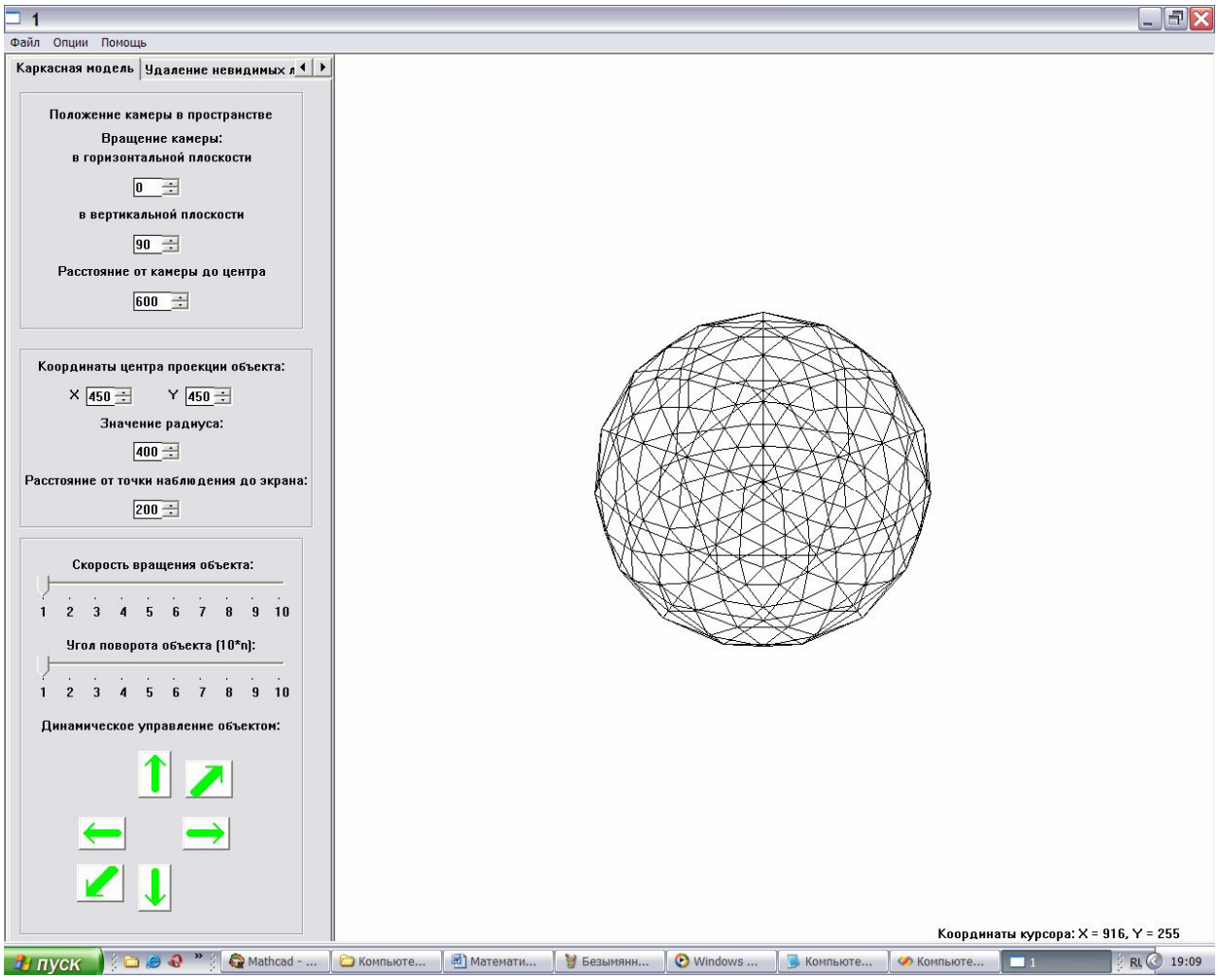


Рисунок 16. Построенная сфера с 320 гранями.

4.5 Удаление невидимых линий объекта

В данной программе удаление невидимых линий объекта реализовано с помощью алгоритма Робертса. Это первый алгоритм по удалению невидимых линий, предложенный Робертсом, требующий, чтобы каждая грань была выпуклым многоугольником.

Описание данного алгоритма

Если грань обращена к наблюдателю внешней стороной, то она видима, так как грани, которые обращены к наблюдателю внутренней стороной будут закрываться гранями, обращенными внешней стороной.

Анализ видимости грани проводится на основе проверки направления обхода проекций трех последовательных вершин грани трехмерного объекта. Если (dx1*dy2-dx2*dy1)>0, то обход осуществляется по часовой стрелки, грань располагается к наблюдателю внутренней стороной, заслоняется внешней гранью и, как следствие, невидима. В противном случае – грань видима. В приведенном выражении:

dx1=x2-x1;
dx2=x3-x2;
dy1=y2-y1;
dy2=y3-y2;

Ребро многогранника невидимо в том случае, если оно лежит на пересечении двух невидимых (нелицевых) граней. Алгоритм функции по реализации удаления невидимых линий, а также его описание представлены в разделе “Основные алгоритмы программы” на странице .

4.6 Закраска

В данной курсовой работе использована однотонная закраска объекта, реализованная с помощью алгоритма построчного сканирования.

Описание данного алгоритма

В каждый момент времени, т.е. при рассмотрении каждой сканирующей строки рассматриваются только те ребра, которые пересекают эту строку. Для хранения информации об этих ребрах используются списки активных ребер, которые формируются в процессе алгоритма.

На каждом шаге в список могут добавляться новые ребра, которые до этого не пересекали строку.

Чтобы сформировать список активных ребер (САР) в начале алгоритма формируется список всех ребер (СР), каждый элемент которого содержит информацию о каждом ребре.

Поля элемента СР:

Ребро 1			
Ребро 2			
…			
ygr	x	dx	dy
…			

“ygr” – значение номера наивысшей сканирующей
строки, пересекающей ребро;
“x” – начальное значение координаты x точки пересечения ребра с наивысшей сканирующей строкой;
“dy”- число сканирующих строк, пересекающих
ребро грани;
“dx” – приращение по оси X при переходе от одной сканирующей строки к другой.

1 шаг. Определить для каждого ребра грани куба наивысшую сканирующую строку. Горизонтальные грани не учитываются.
2 шаг. Занести информацию о ребрах в СР.
3 шаг. Занести в САР информацию о ребрах с наибольшим значением поля ygr.
4 шаг. Отсортировать САР в порядке возрастания поля x.
5 шаг. Выделить пары элементов САР.
6 шаг. Для каждой пары элементов САР активировать на сканирующей строке y=”CAP.ygr” заданным цветом закраски пиксели для целых значений x, таких что CAP[j].x<=x<=CAP[j+1].x (рисунок 17). Для каждого ребра (i= 1, k) из САР уменьшить значение полей “dy” и “ygr” на 1. Если CAP[i].dy<0, то исключить данное ребро из САР. Вычислить новое значение полей “x” элементов САР:

$$CAP[i].x_{нов}=CAP[i].x_{стар}+CAP[i].dx.$$

7 шаг. Добавить в САР элементы из списка ребер, у которых поле ygr=y-1.
8 шаг. Если САР не пуст, перейти к шагу 4.
9 шаг. Конец алгоритма.

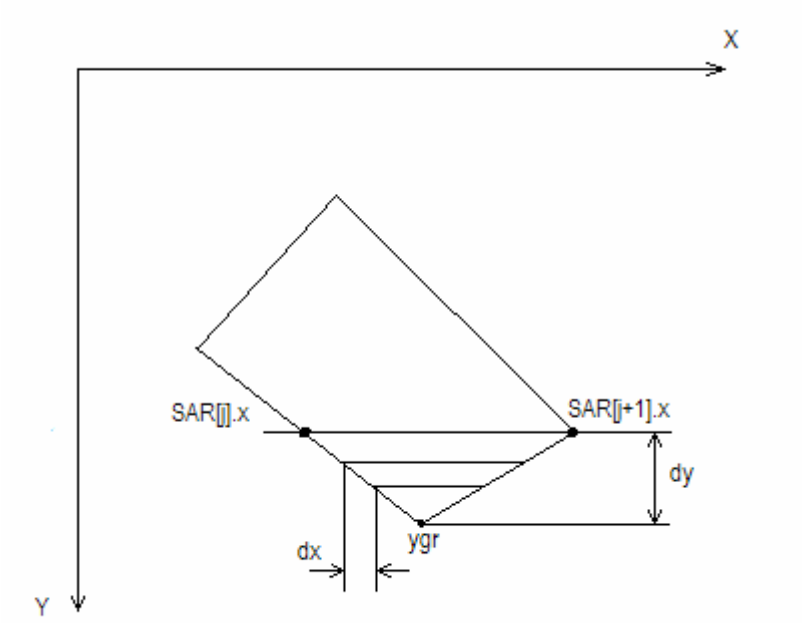


Рисунок 17. Параметры применительно к изображению.

Алгоритм функции по реализации закраски сферы, а также его описание представлены в разделе “Основные алгоритмы программы” на странице .

5. *Структура программы*

Структура программы изображена на рисунке 18.

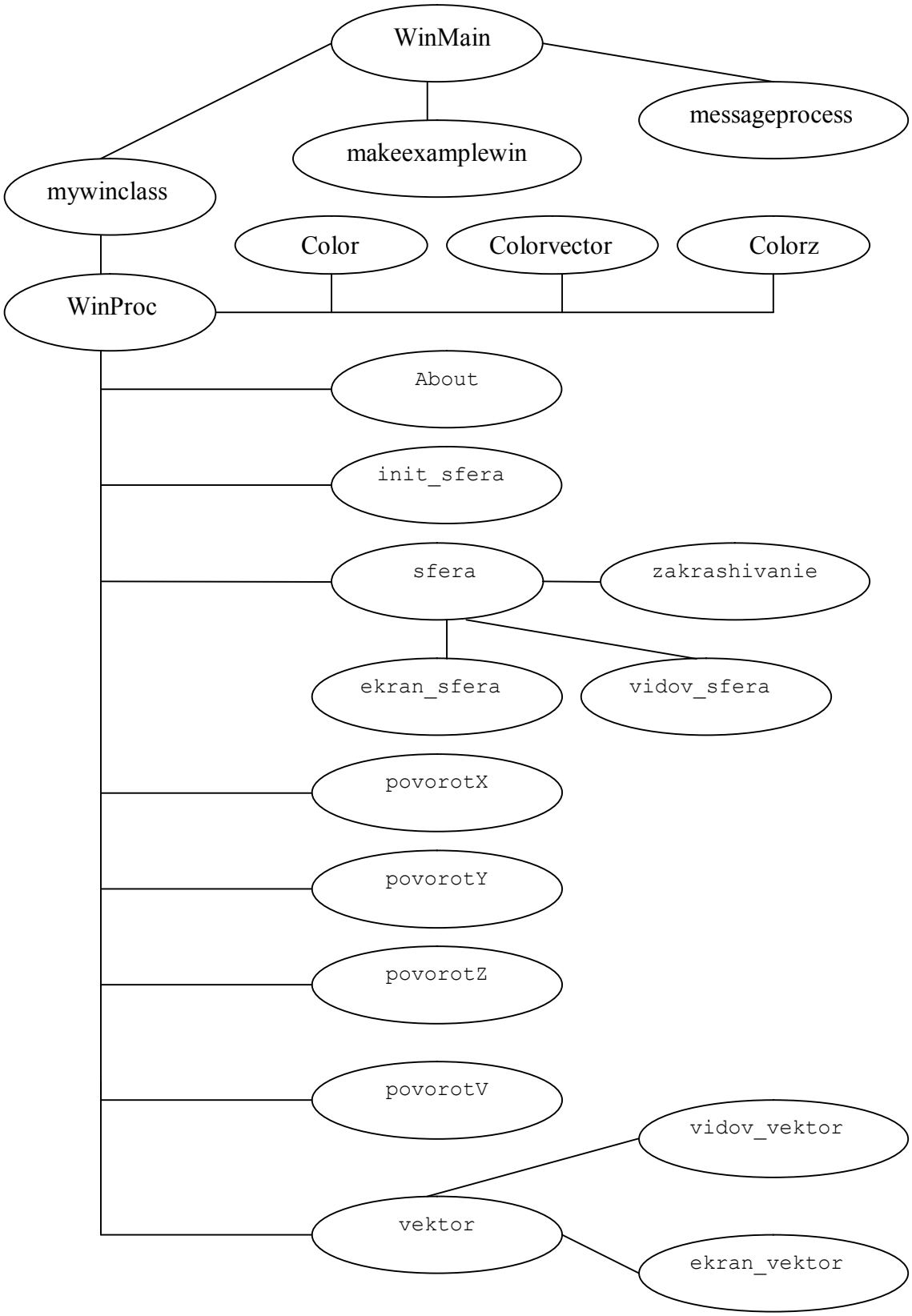


Рисунок 18. Структура программы.

Все функции, определяющие построение сферы и ее изображение на экране вызываются из функции обработки сообщений WinProc. Это следующие функции:
- init_sfera – инициализация координат сферы в мировой системе координат. Функция не имеет параметров и ничего не возвращает.
- sfera – построение сферы по проинициализированным координатам, преобразование мировых координат в видовые (за счет вызова функции vidov_sfera), а также видовых в экранные (за счет вызова функции ekran_sfera), удаление невидимых линий (по требованию пользователя), вызов функции закрашивания сферы (по требованию пользователя). Алгоритм данной функции представлен в разделе "Основные алгоритмы программы" на страницах 23-26.
- povorotX – поворот сферы на заданный угол вокруг оси X. Функция имеет один параметр – переменную s, назначение которой определение вращения сферы (по часовой или против часовой) и ничего не возвращает.
- povorotY – поворот сферы на заданный угол вокруг оси Y. Функция имеет один параметр – переменную s, назначение которой определение вращения сферы (по часовой или против часовой) и ничего не возвращает.
- povorotZ – поворот сферы на заданный угол вокруг оси Z. Функция имеет один параметр – переменную s, назначение которой определение вращения сферы (по часовой или против часовой) и ничего не возвращает.
- povorotV – поворот сферы на заданный угол вокруг произвольного вектора. Функция имеет один параметр – переменную s, назначение которой определение вращения сферы (по часовой или против часовой) и ничего не возвращает.
- vektor – инициализация и построение вектора, вокруг которого осуществляется поворот сферы, преобразование мировых координат вектора в видовые (за счет вызова функции vidov_vektor), видовых в экранные (за счет вызова функции ekran_vektor). Функция имеет два параметра: HDC hdc – контекст графического устройства (экран) для вывода сферы в среде Windows; HPEN hpen – созданное ранее перо, которым будут рисоваться ребра сферы. Данная функция также ничего не возвращает.
- vidov_vector – преобразование мировых координат вектора в видовые. В нее передается переменная i, определяющая номер координаты вектора, в массивах инициализации. Функция ничего не возвращает,
- ekran_vector – преобразование видовых координат вектора в экранные. В нее передается переменная i, определяющая номер координаты вектора, в массивах инициализации. Функция ничего не возвращает,
- vidov_sfera – преобразование мировых координат сферы в видовые. Алгоритм данной функции представлен в разделе "Основные алгоритмы программы" на страницах 27.
- ekran_sfera – преобразование видовых координат сферы в экранные. Алгоритм данной функции представлен в разделе "Основные алгоритмы программы" на страницах 27-28.
- zakrashivanie – закрашивает сферу. Вызывается из функции sfera. Имеет один параметр HDC hdc – контекст графического устройства (экран) для вывода сферы в среде Windows. Алгоритм данной функции представлен в разделе "Основные алгоритмы программы" настраницах 28-33.

Функция WinProc, фактически, представляет собой оконную процедуру. Она обрабатывает все сообщения, посланные от мыши, элементов управления и др. в программе.
Остальные функции программы:
- About – функция, обрабатывающая сообщения от диалогового окна "О программе";
- Color – заполнение структуры определения цветовой палитры для выбора цвета ребра сферы;
- Colorvector – заполнение структуры определения цветовой палитры для выбора цвета вектора;

- ColorZ – заполнение структуры определения цветовой палитры для выбора цвета заполнения сферы;
- mywinclass – регистрация класса окна;
- makeexamplewin – создание экземпляра приложения – непосредственно главного окна программы;
- messageprocess – обработчик цикла сообщений.

Глобальные переменные, структуры, дескрипторы и массивы представлены в таблице 1.

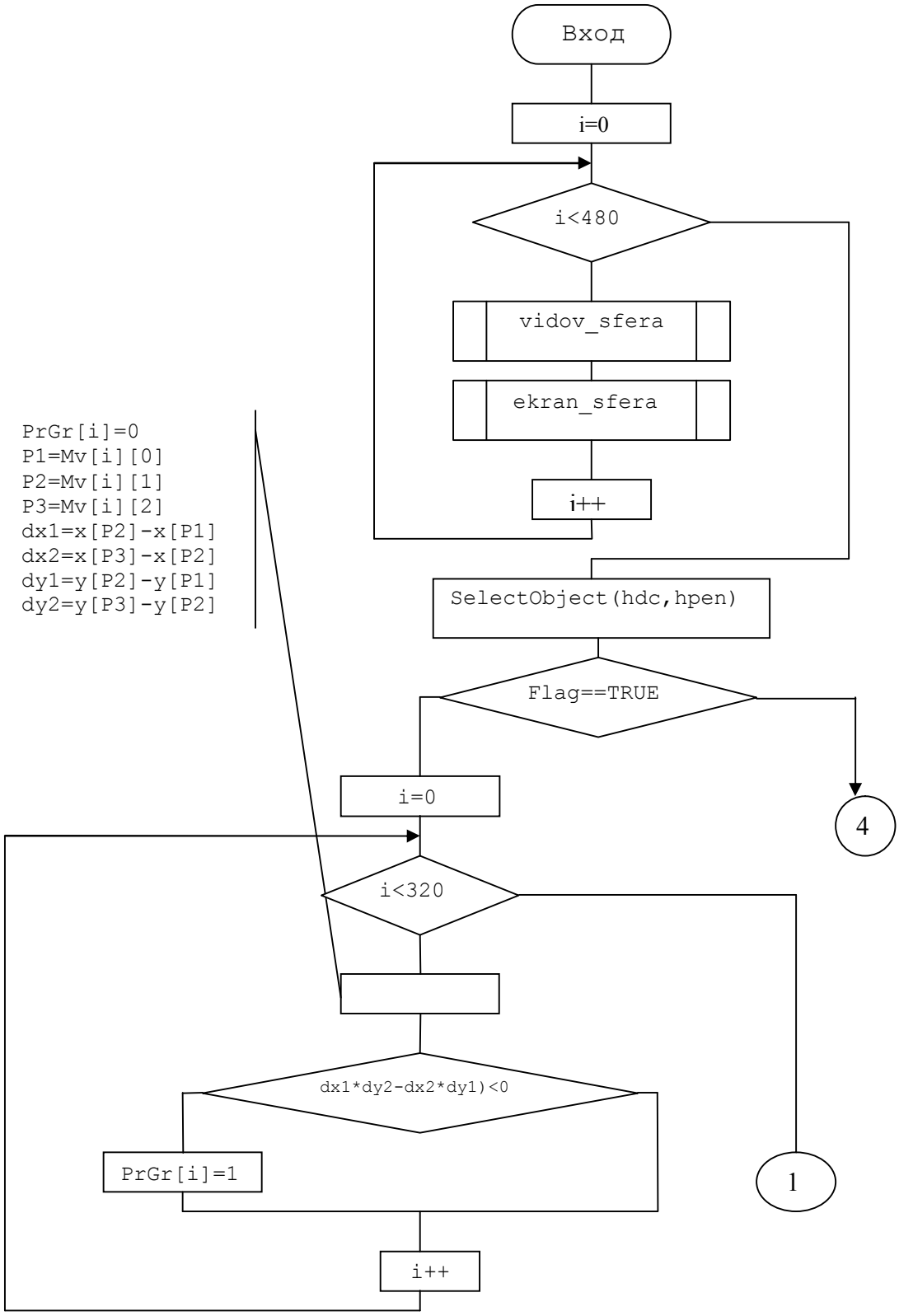
Название	Тип	Назначение
app	HINSTANCE	идентификатор приложения
hwTabC	HWND	дескриптор окна-закладки
crColor	COLORREF	Структура передает цвет для ребра сферы
crColorZ	COLORREF	Структура передает цвет для заполнения сферы
crColorvector	COLORREF	Структура передает цвет для вектора
Flag	BOOL	Флаг для удаление невидимых линий
Flag2	BOOL	Флаг для удаления вектора
Flagz	BOOL	Флаг для удаления закраски
Flagpr	BOOL	Флаг для смены цвета закраски
Flagc	BOOL	Флаг для смены проекции сферы
kvx	int	Координата первой точки вектора по x
kvy	int	Координата первой точки вектора по y
kvz	int	Координата первой точки вектора по z
kvx2	int	Координата второй точки вектора по x
kvy2	int	Координата второй точки вектора по y
kvz2	int	Координата второй точки вектора по z
xx	int	Координата центра проекции объекта по x
yy	int	Координата центра проекции объекта по y
chast=1	int	Для изменения угла поворота сферы
skor=1	int	Для изменения скорости вращения сферы
a	float	Радиус сферы
d	float	Расстояние от центра проецирования до проекционной плоскости
tetta	float	Угол отклонения проекции точки наблюдения на плоскость XY от оси x
phi	float	Угол отклонения вектора наблюдения от оси z
R0	float	Расстояние от точки наблюдения до начала мировой системы

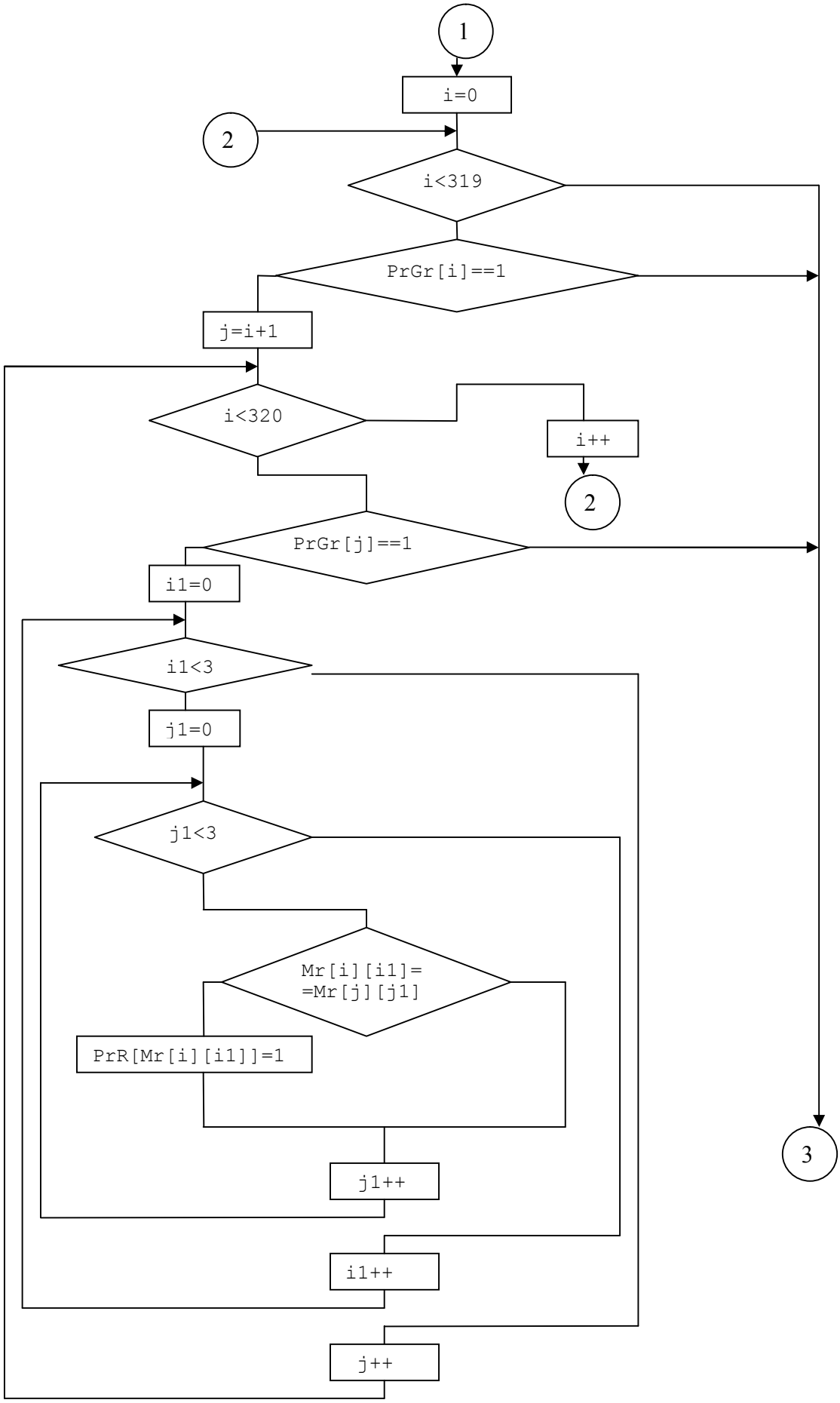
		координат
i	int	Переменная, использующаяся в циклах программы
p=1	float	Для перевода в градусную меру углов поворота камеры
rec	RECT	Структура, определяющая границы области окна для вывода изображения
nTab	int	Определяет элемент управления от которого пришло сообщение
lpToolTipText	LPTOOLTIPTEXT	Структура, определяющая сообщение после действия с закладкой
lpNMHdr	LPNMHDR	Передает реакцию на выбор закладки
TC_Item	TC_ITEM	Структура для создания окна закладки
PaintStruct	PAINTSTRUCT	Структура, содержащая информацию для обновления клиентской области окна
ptCursor	POINT	Содержит координаты x, y точки
hDC, hhdc, hhdcWin	HDC	Дескрипторы контекстов устройств
hhBitmap	HBITMAP	Дескриптор изображения
hPen, hPenv	HPEN	Дескрипторы перьев
xW[182],yW[182],zW[182]	float	Мировые координаты сферы
xV[182],yV[182],zV[182]	float	Видовые координаты сферы
x[182],y[182]	float	Экранные координаты сферы
xWv[2],yWv[2],zWv[2]	float	Мировые координаты вектора
xVv[2],yVv[2],zVv[2]	float	Видовые координаты вектора
xv[2],yv[2]	float	Экранные координаты вектора
PrGr[320]	int	Проверочный массив по граням
PrR[480]	int	Проверочный массив по ребрам
El[50],F[20],G[30]	HWND	Массивы для вывода и удаления с поля зрения управляющих элементов на закладках
Mr[320][3]	int	Массив граней по реберам
Mv[320][3]	int	Массив граней по вершинам
K[480][2]	int	Массив ребер
szMsg[128]	char	Массив для вывода на окно координат мыши

Таблица 1. Глобальные переменные, структуры, дескрипторы и массивы программы.

6. Основные алгоритмы программы

Алгоритм функция sfera. Вызывается в функции обработки сообщений WinProc. Данный алгоритм представлен на рисунке 19.




1
i=0
2
i<319
PrGr[i]==1
j=i+1
i<320
i++
2
PrGr[j]==1
i1=0
i1<3
j1=0
j1<3
Mr[i][i1]=
=Mr[j][j1]
PrR[Mr[i][i1]]=1
j1++
i1++
j++
3

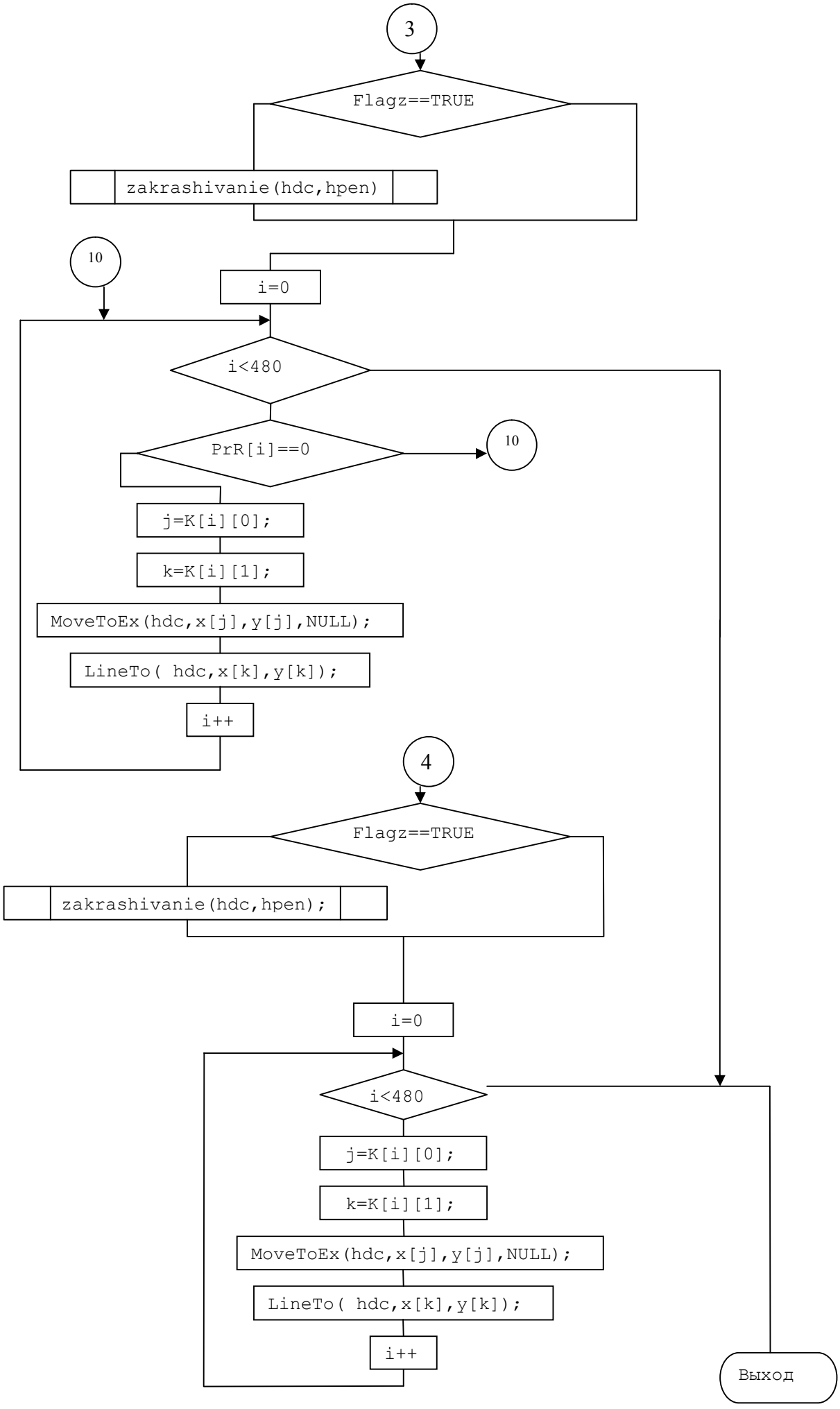

3
Flagz==TRUE
zakrashivanie(hdc,hpen)
10
i=0
i<480
PrR[i]==0
10
j=K[i][0];
k=K[i][1];
MoveToEx(hdc,x[j],y[j],NULL);
LineTo(hdc,x[k],y[k]);
i++
4
Flagz==TRUE
zakrashivanie(hdc,hpen);
i=0
i<480
j=K[i][0];
k=K[i][1];
MoveToEx(hdc,x[j],y[j],NULL);
LineTo(hdc,x[k],y[k]);
i++
Выход

Рисунок 19. Алгоритм функции sfera.

Локальные переменные функции sfera представлены в таблице 2.

Название	Тип	Назначение
k	int	Для вывода ребер сферы
j	int	Для вывода ребер сферы
P1	int	Для определения первой точки на грани
P2	int	Для определения второй точки на грани
P3	int	Для определения третьей точки на грани
dx1	float	Разница x2-x1
dx2	float	Разница x3-x2
dy1	float	Разница y2-y1
dy2	float	Разница y3-y2

Таблица 2. Локальные переменные функции sfera.

В программе функция выглядит как: `void sfera (HDC hdc,HPEN hpen){…}`
В функцию передаются два параметра:

1 – HDC hdc – контекст графического устройства (экран) для вывода сферы в среде Windows;
2 – HPEN hpen – созданное ранее перо, которым будут рисоваться ребра сферы.

Функция ничего не возвращает, тип – void. В ней же вызываются три функции: vidov_sfera, ekran_sfera, zakrashivanie.

Пояснения к алгоритму:

В начале алгоритма в цикле заполняется нулями проверочный массив по ребрам, после вызываются функции vidov_sfera, ekran_sfera в цикле 182 раза (по количеству проинициализированных в функции init_sfera координат). Далее в цикле производится определение параметров dx1, dx2, dy1, dy2 и выполняется проверка выражения (dx1*dy2-dx2*dy1) – определяется направление обхода грани (см. в раздел Математическая реализация задания ->Удаление невидимых линий объекта). Далее проводится поиск невидимых граней. Если грани не видимы, то их общие ребра также невидимы. После этого, выводятся только видимые ребра. В алгоритме присутствуют 2 флага, объявленные глобально (описание приведено в таблице 1). Первый флаг – Flag, необходим для ответа системы на требование пользователя удалить невидимые ребра (если значение флага равно FALSE, не реализуется блок алгоритма, отвечающий за удаление невидимых ребер). Второй флаг – Flagz необходим для ответа системы на требование пользователя закрасить сферу (если значение флага равно TRUE, вызывается функция закраски zakrashivanie).

2. Алгоритм функции vidov_sfera. Функция вызывается в функции sfera. Данный алгоритм представлен на рисунке 20.

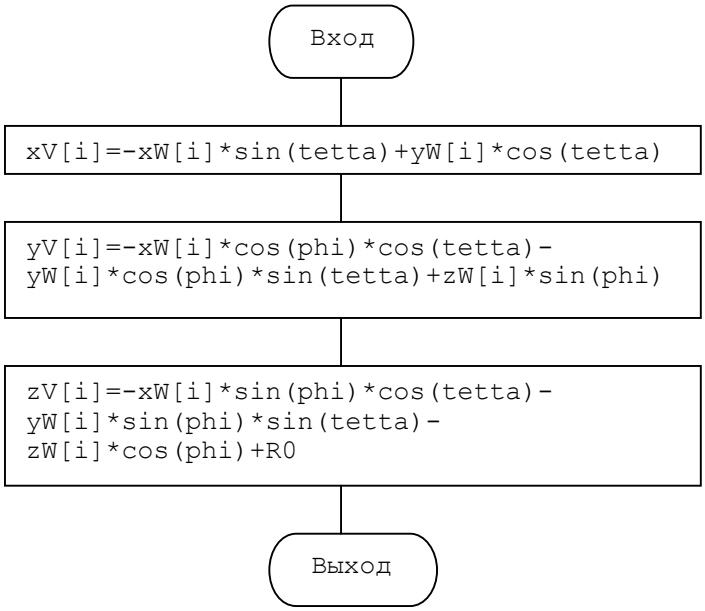


Рисунок 20. Алгоритм функции vidov_sfera.

Функция необходима для перевода мировых координат объекта в видовые координаты, так чтобы начало видовой системы координат совпало с точкой наблюдения. Данная функция вызывается в функции рисования сферы sfera.

В программе функция выглядит как: `void vidov_sfera(int i){…}`
В нее передается переменная i, определяющая номер координаты сферы, в массивах инициализации. Функция ничего не возвращает, тип – void.

3. Алгоритм функции ekran_sfera. Функция вызывается в функции sfera. Данный алгоритм представлен на рисунке 21.

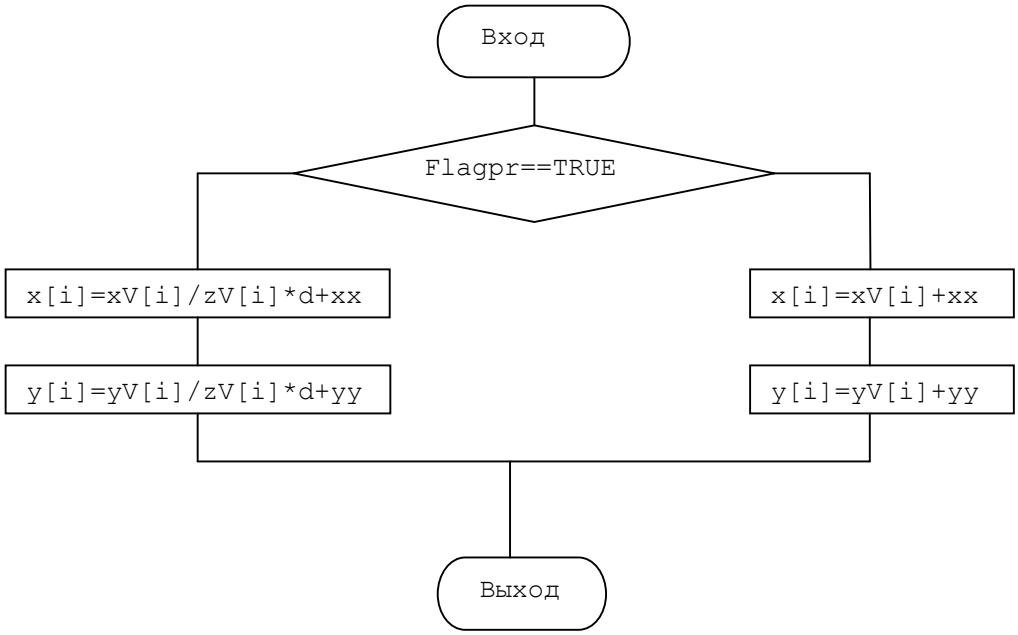
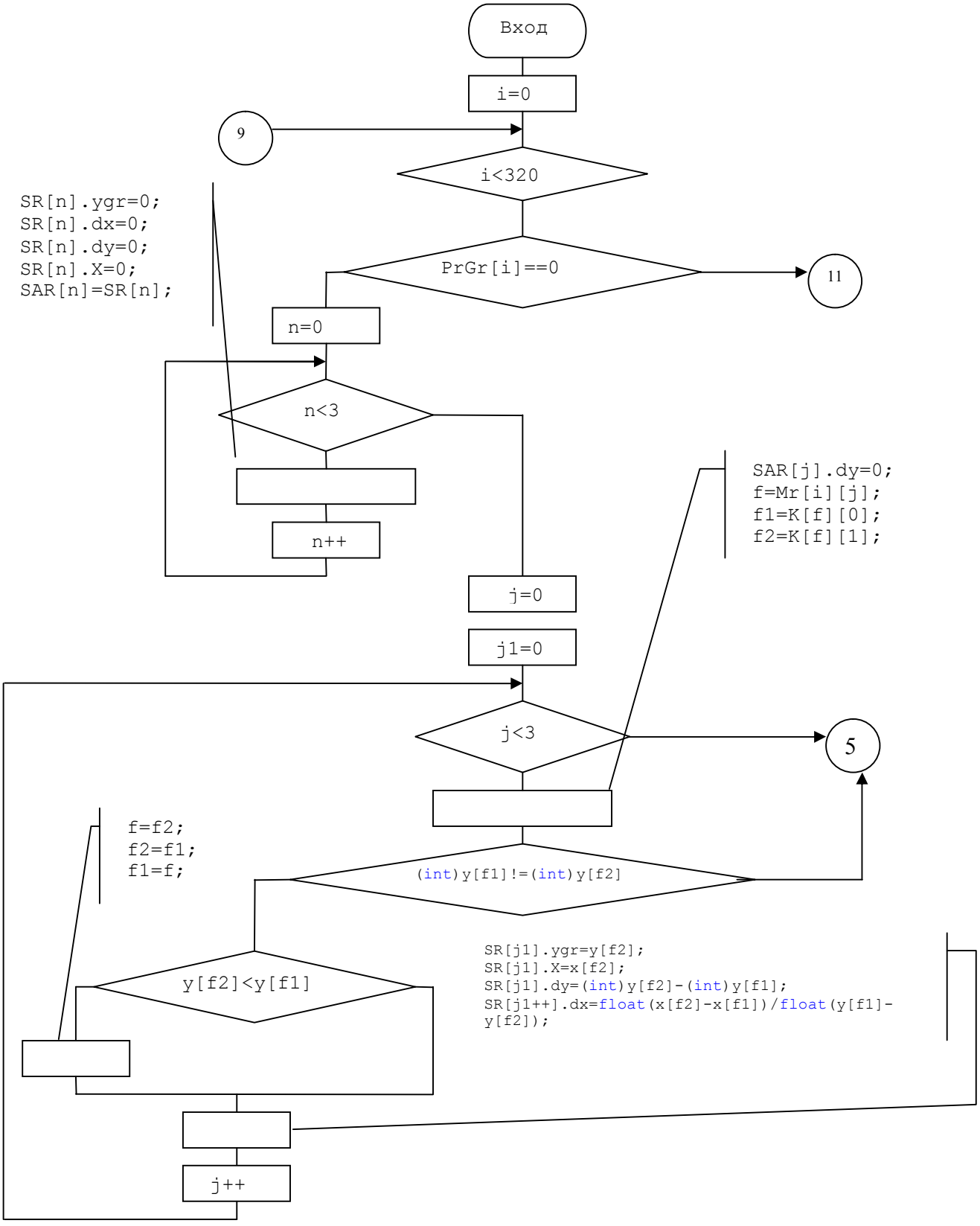


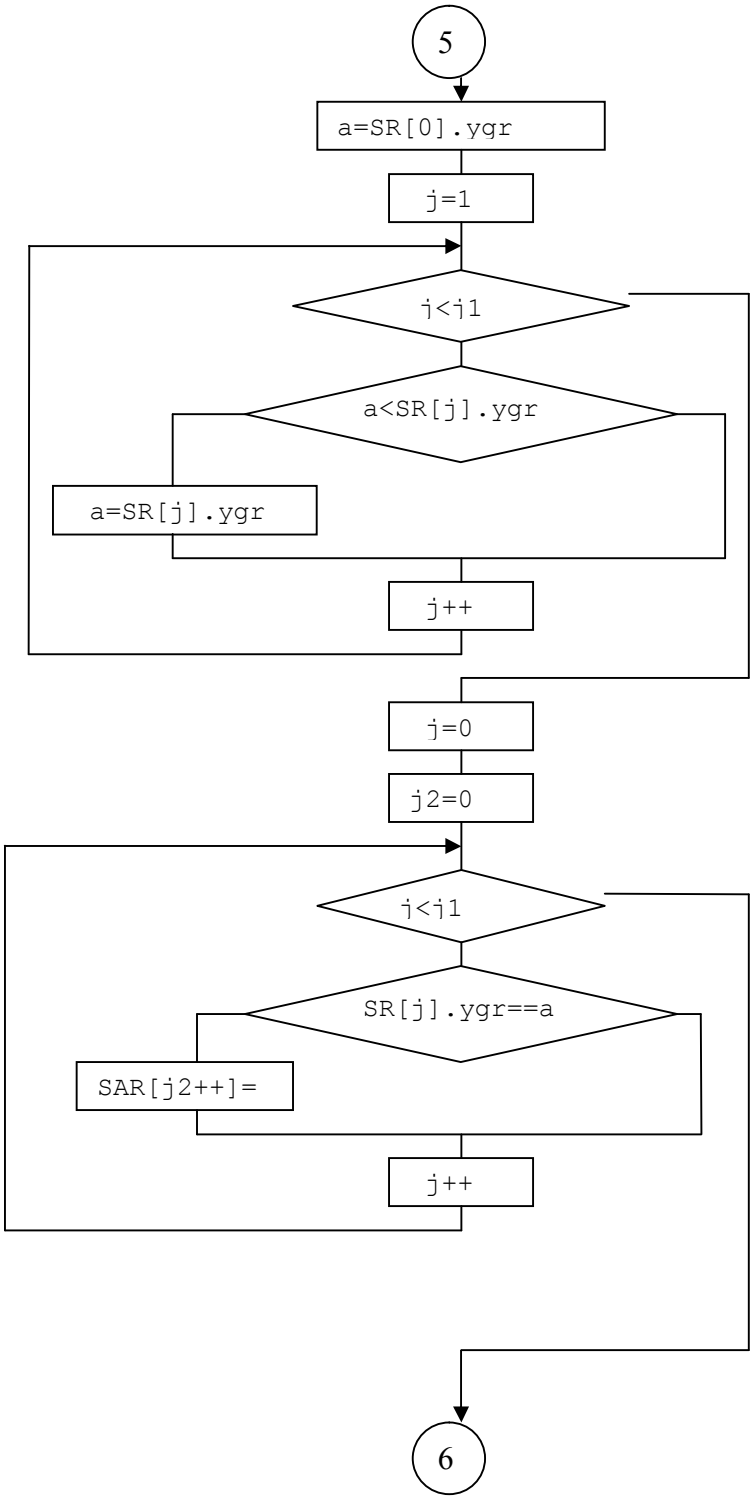
Рисунок 21. Алгоритм функции ekran_sfera.

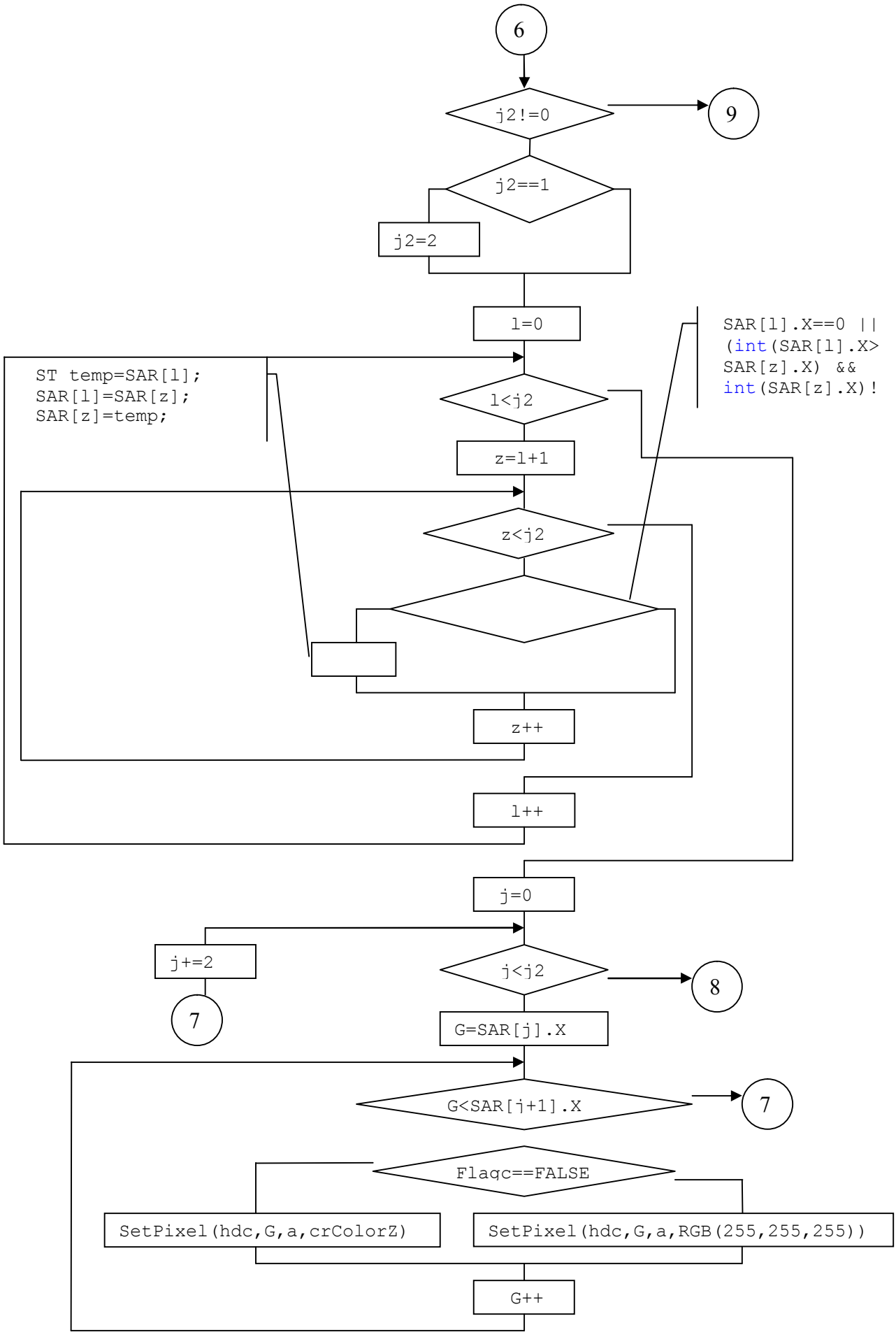
Функция необходима для перевода видовых координат объекта в экранные координаты для того, чтобы представить изображение объекта не в трехмерном пространстве, а на двумерной плоскости. Данная функция вызывается в функции рисования сферы sfera.

В программе функция выглядит как: `void ekran_sfera(int i){…}`
В нее передается переменная i, определяющая номер координаты сферы, в массивах инициализации. Функция ничего не возвращает, тип – void.

Алгоритм функции zakrashivanie. Функция вызывается в функции sfera. Данный алгоритм представлен на рисунке 22.


Вход
i=0
9
i<320
SR[n].ygr=0;
SR[n].dx=0;
SR[n].dy=0;
SR[n].X=0;
SAR[n]=SR[n];
PrGr[i]==0
11
n=0
n<3
n++
SAR[j].dy=0;
f=Mr[i][j];
f1=K[f][0];
f2=K[f][1];
j=0
j1=0
j<3
5
f=f2;
f2=f1;
f1=f;
(int)y[f1]!=(int)y[f2]
y[f2]<y[f1]
SR[j1].ygr=y[f2];
SR[j1].X=x[f2];
SR[j1].dy=(int)y[f2]-(int)y[f1];
SR[j1++].dx=float(x[f2]-x[f1])/float(y[f1]-y[f2]);
j++


5
a=SR[0].ygr
j=1
j<j1
a<SR[j].ygr
a=SR[j].ygr
j++
j=0
j2=0
j<j1
SR[j].ygr==a
SAR[j2++]=
j++
6


6
j2!=0
9
j2==1
j2=2
l=0
SAR[l].X==0 ||
(int(SAR[l].X>
SAR[z].X) &&
int(SAR[z].X)!
ST temp=SAR[l];
SAR[l]=SAR[z];
SAR[z]=temp;
l<j2
z=l+1
z<j2
z++
l++
j=0
j+=2
j<j2
8
7
7
G=SAR[j].X
G<SAR[j+1].X
7
Flagc==FALSE
SetPixel(hdc,G,a,crColorZ)
SetPixel(hdc,G,a,RGB(255,255,255))
G++

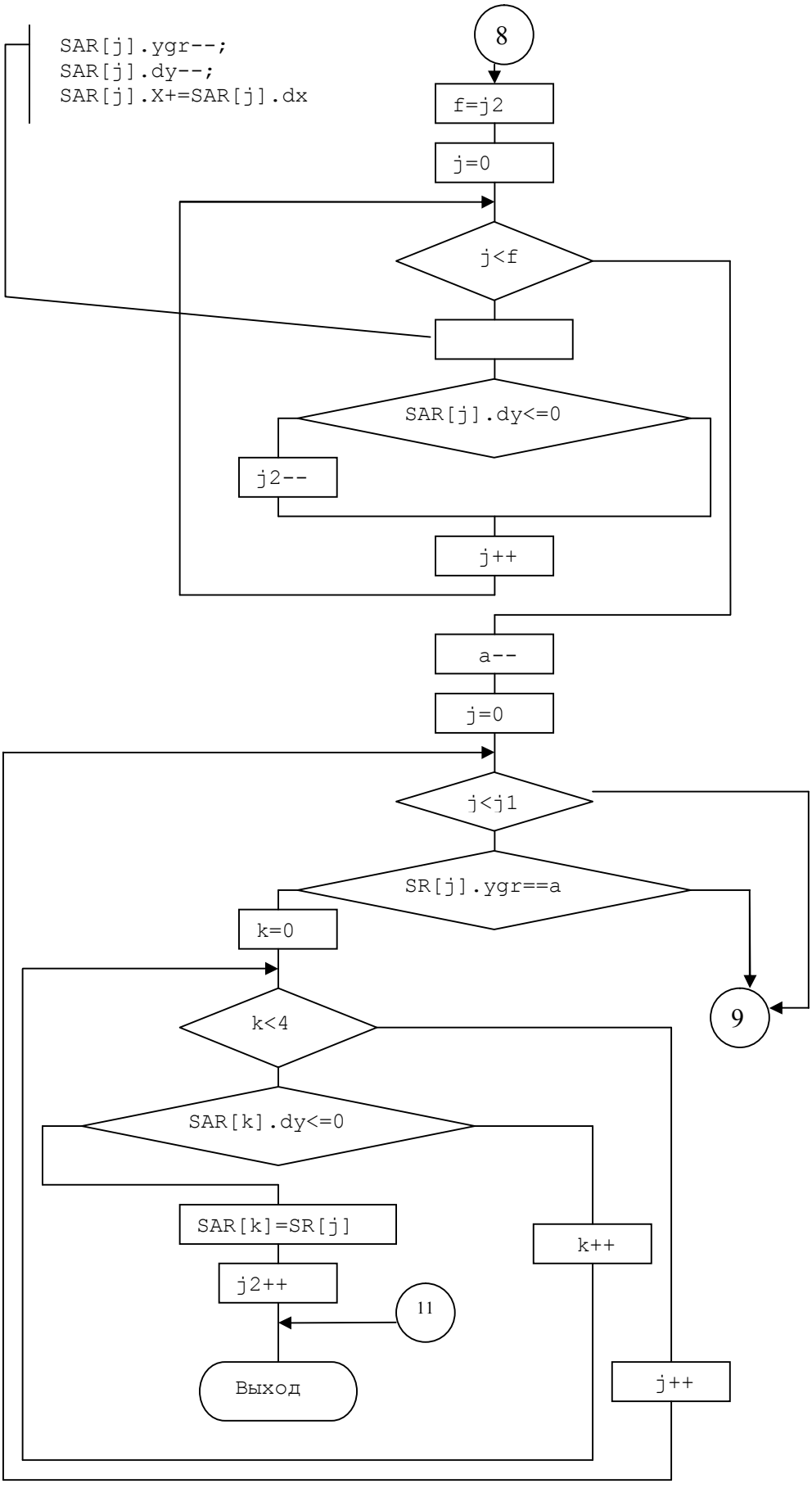

SAR[j].ygr--;
SAR[j].dy--;
SAR[j].X+=SAR[j].dx
8
f=j2
j=0
j<f
SAR[j].dy<=0
j2--
j++
a--
j=0
j<j1
SR[j].ygr==a
k=0
9
k<4
SAR[k].dy<=0
SAR[k]=SR[j]
k++
j2++
11
Выход
j++

Рисунок 22. Алгоритм функции zakrashivanie.

Локальные переменные и структуры функции zakrashivanie представлены в таблице 3.

Название	Тип	Назначение
`SR[3]`	`struct ST`	Список ребер
`SAR[3]`	`struct ST`	Список активных ребер
f	int	Для определения ребра
f1	int	Для определения первой точки ребра
f2	int	Для определения второй точки ребра

Таблица 3. Локальные переменные и структуры функции zakrashivanie.

В программе функция выглядит как: `void zakrashivanie(HDC hdc){…}`

В функцию передаются параметр – HDC hdc – контекст графического устройства (экран) для вывода закраски сферы в среде Windows. Функция ничего не возвращает, тип – void.

Пояснения к алгоритму:

В начале алгоритма объявляются две структуры, составляющих список ребер и список активных ребер. После производится поиск не горизонтальных ребер. Далее алгоритм функционирует так, как описано в разделе математическая реализация задания -> Закраска. В алгоритме присутствует флаг – `Flagc`, описанный в таблице 1. Он определяет цвет, которым будет закрашиваться сфера – цветом фона или выбранным пользователем (необходимо для вращения сферы в пространстве).

7. Руководство пользователя

После запуска программы на исполнение Вы увидите окно, в котором уже изображена сфера в параллельном ортогональном проецировании как показано на рисунке 23.

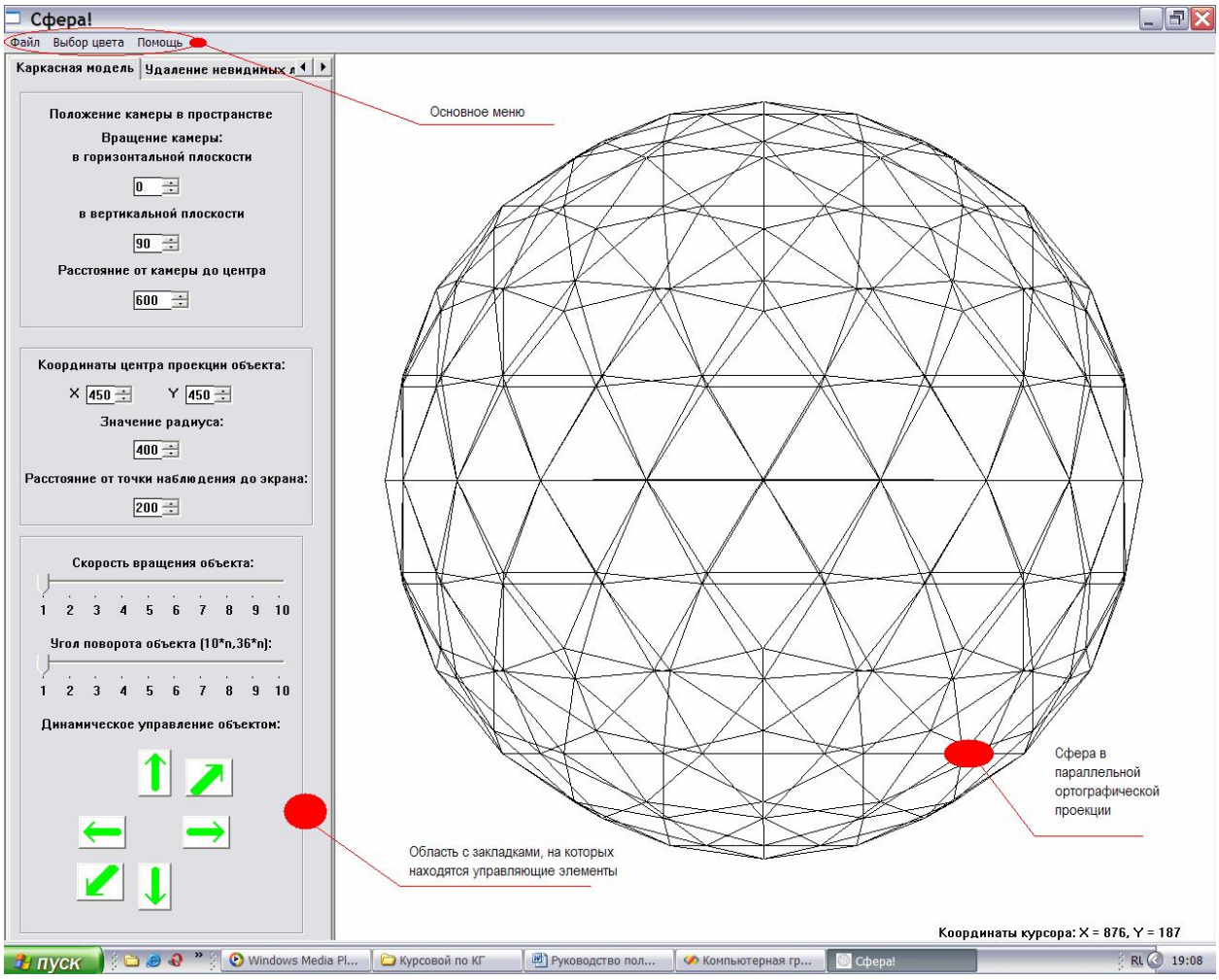


Рисунок 23. Изначальное изображение программы на мониторе.

В данном окне Вы может увидеть саму сферу в ортогональной параллельной проекции, основное меню, состоящее из пунктов “Файл”, “Выбор цвета”, “Помощь”, а также область с закладками, на которых находятся элементы управления.

Область с закладками включает в себя три вкладки:

- каркасная модель;
- удаление невидимых линий, закраска;
- вектор;

Итак, первая область с управляющими элементами – “каркасная модель”. На ней располагаются три блока управления параметрами сферы и ее отображения в пространстве:

- положение камеры в пространстве;

- управление вращением объекта относительно центра масс;
- остальные параметры;

Вы можете устанавливать все параметры во время выполнения программы и следить за тем, как изменяется объект и его изображение.

Установим радиус сферы 120, координаты центра проекции объекта по X:700, по Y:200 (при вводе данных координат можно воспользоваться определителем координат курсора при перемещении мыши в пространстве, все, что необходимо – это просто навести мышь на нужное место, запомнить показанные внизу координаты и ввести их в поля определения центра проекции объекта, значение расстояния от точки наблюдения до экрана оставим без изменений, так как оно используется при центральном одноточечном проецировании, которое устанавливается на вкладки “удаление невидимых линий, закраска”. На рисунке 24 показаны установки данных параметров, а также результат – вывод объекта с измененным радиусом и в другой части экрана.

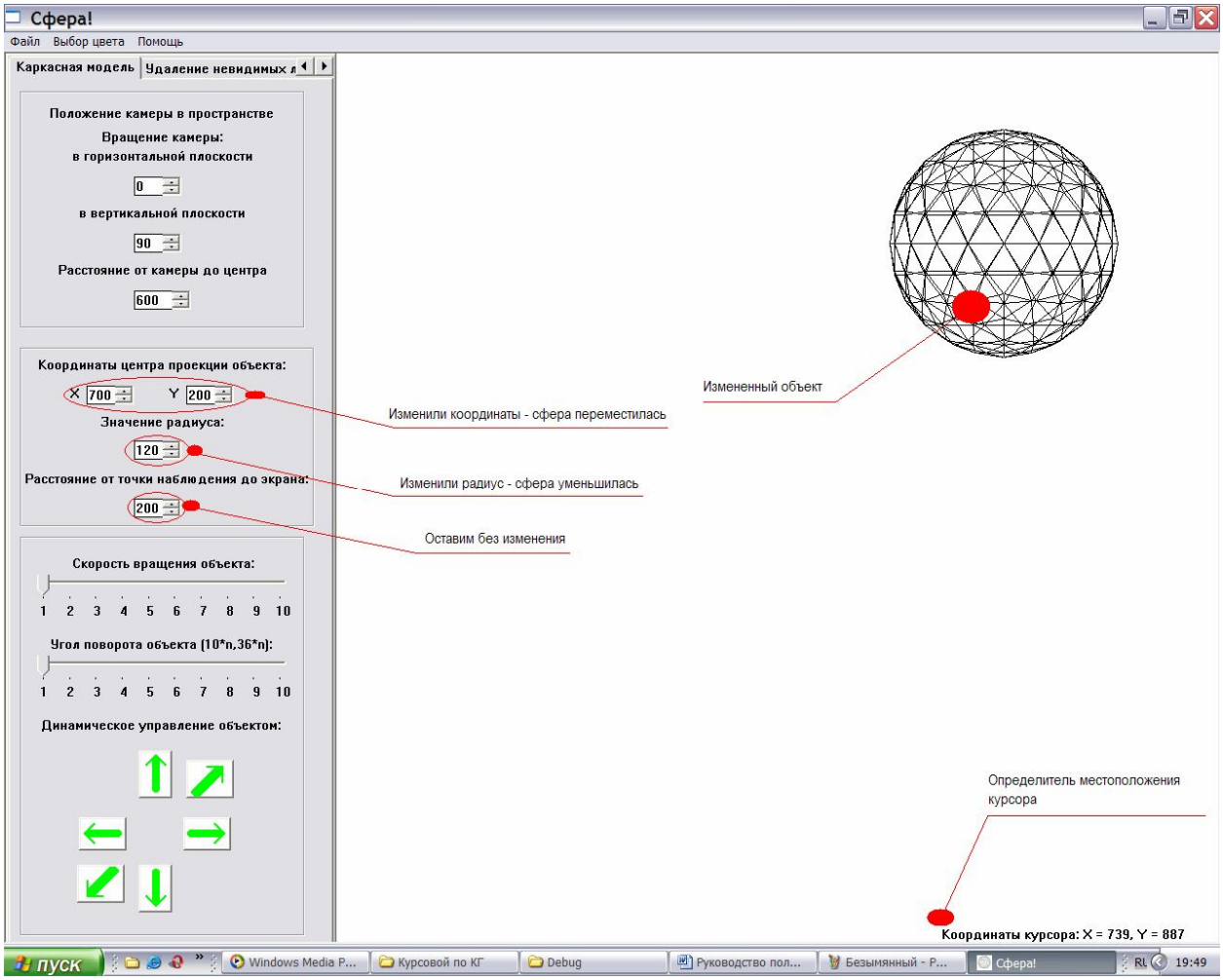


Рисунок 24. Результат установки параметров.

Теперь, давайте, попробуем повращать объект. Внизу данной вкладки Вы видите блок “управления объектом”. Поставим скорость вращения объекта на максимум, то есть на отметку 10, угол поворота объекта на 5 (если вращение объекта происходит вокруг центра масс, то одно деление данной шкалы соответствует повороту объекта на 10 градусов, если вокруг произвольного вектора, параметры которого устанавливаются на

вкладке “вектор”, то цена деления равна 36 градусам). Под делениями вы видите шесть зеленых кнопок, которые показывают вращение объекта по направлениям, указываемыми стрелками. Установки параметров управления объектом и результат его поворота показаны на рисунке 25.

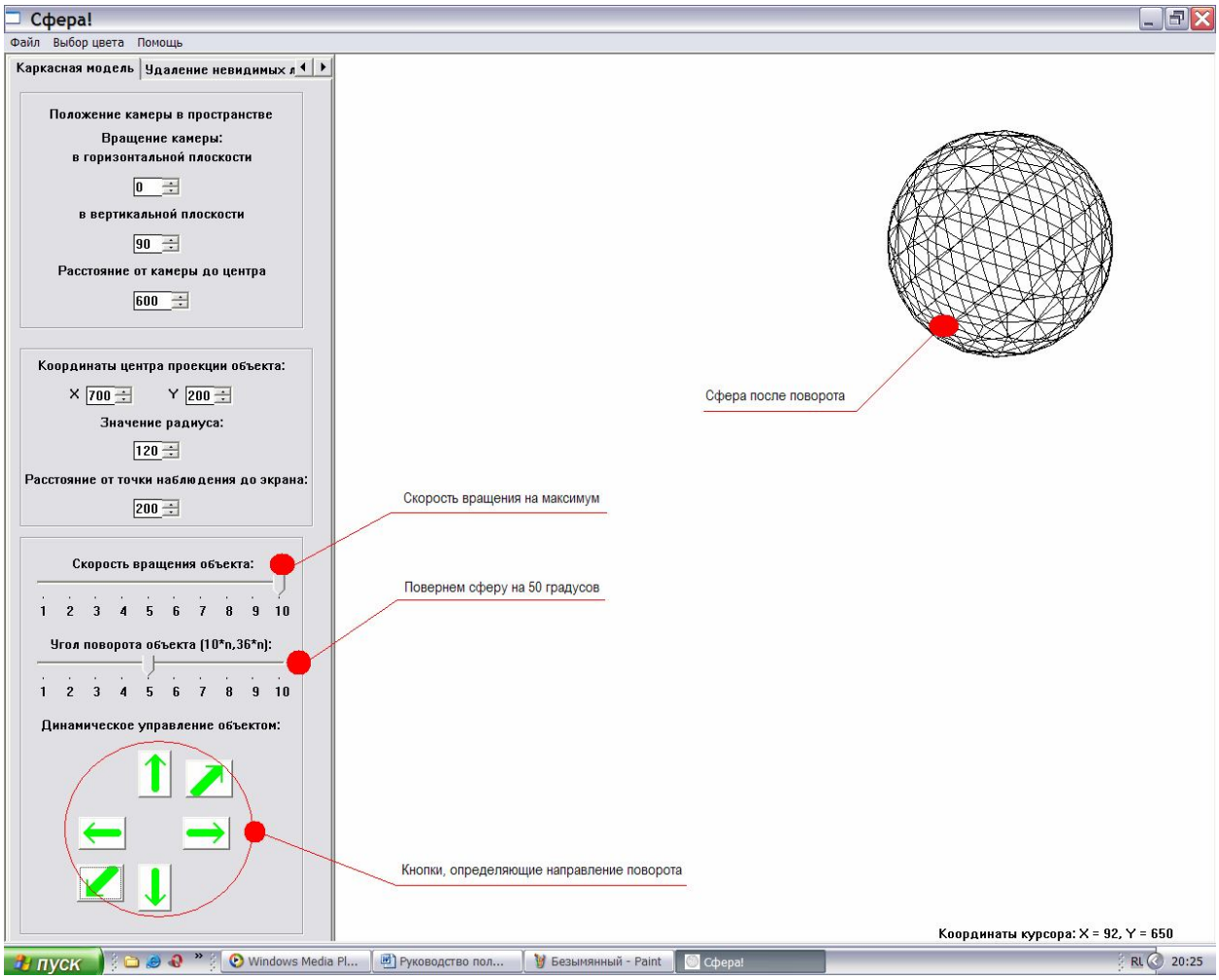


Рисунок 25. Результат поворота сферы.

Параметры, определяющие положение камеры в пространстве, изменим, после того как на вкладке “вектор” установим “Показать вектор”. Это необходимо для того, чтобы увидеть результат изменения положения камеры.

Перейдем на вкладку “удаление невидимых линий, закраска”. Удалим невидимые линии объекта. Установим переключатель “Невидимые линии объекта: удалить”. На рисунке 26 представлен результат данной операции.

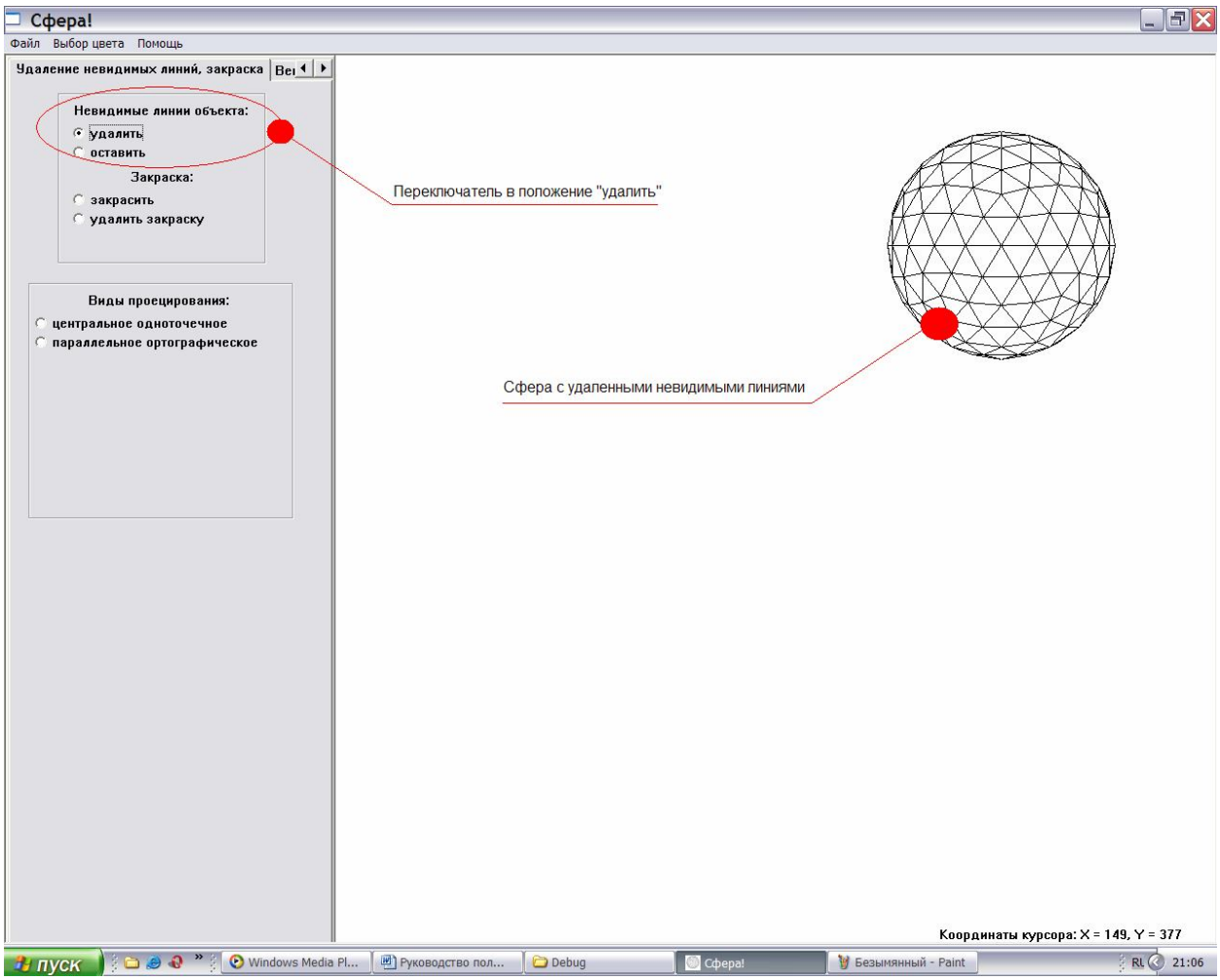


Рисунок 26. Сфера с удаленными невидимыми линиями.

Теперь закрасим объект. Для этого установим переключатель “Закраска: закрасить”. На рисунке 27 представлен результат данной операции.

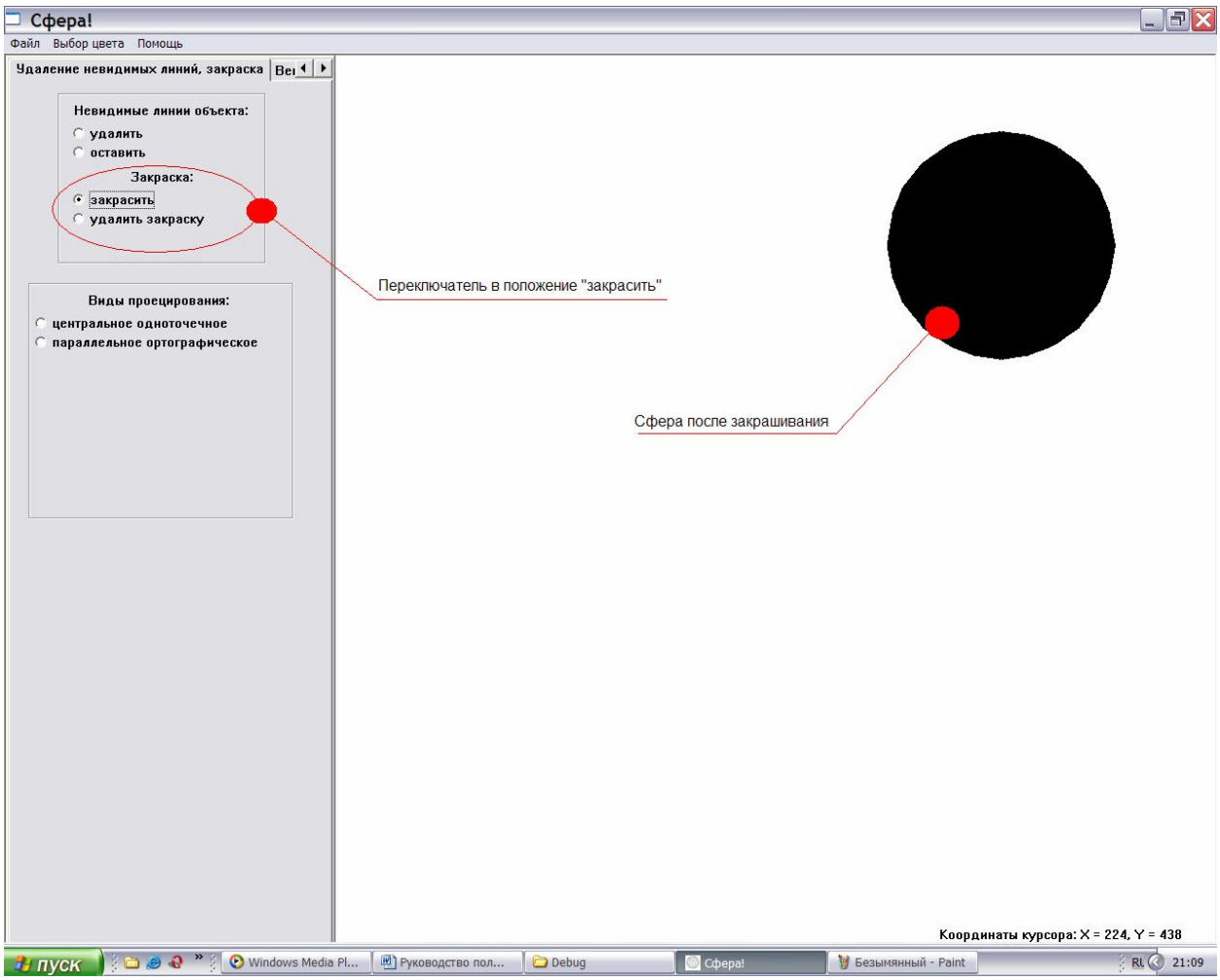


Рисунок 27. Закрашенная сфера.

По умолчанию и ребра, и закраска сферы установлены в черный цвет. Давайте его поменяем. В основном меню выберем пункт “Выбор цвета” и установим цвет для ребер сферы и для ее закраски как на рисунке 28.

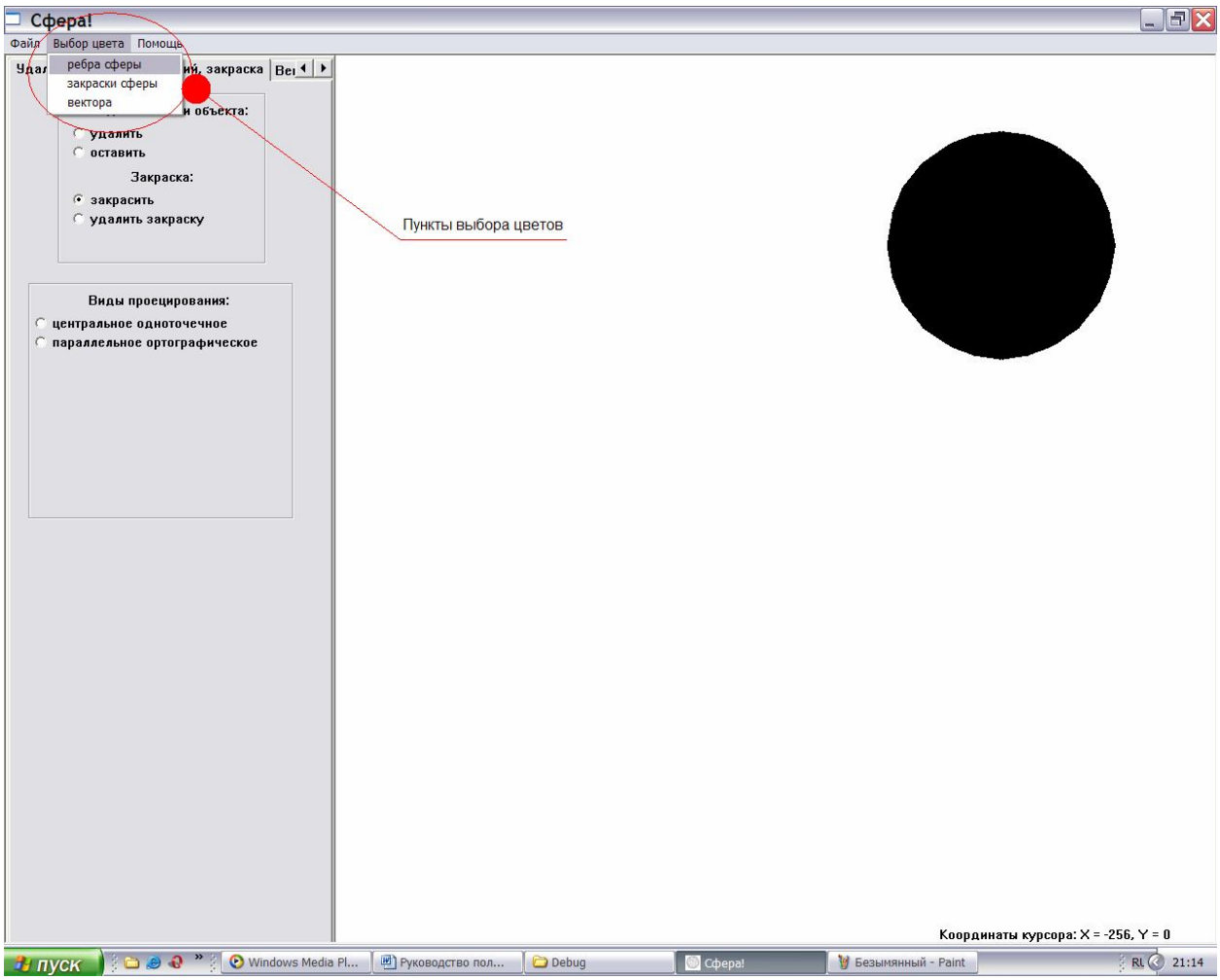


Рисунок 28. Показ пунктов выбора цветов.

Результат выбора цветов показан на рисунке 29.

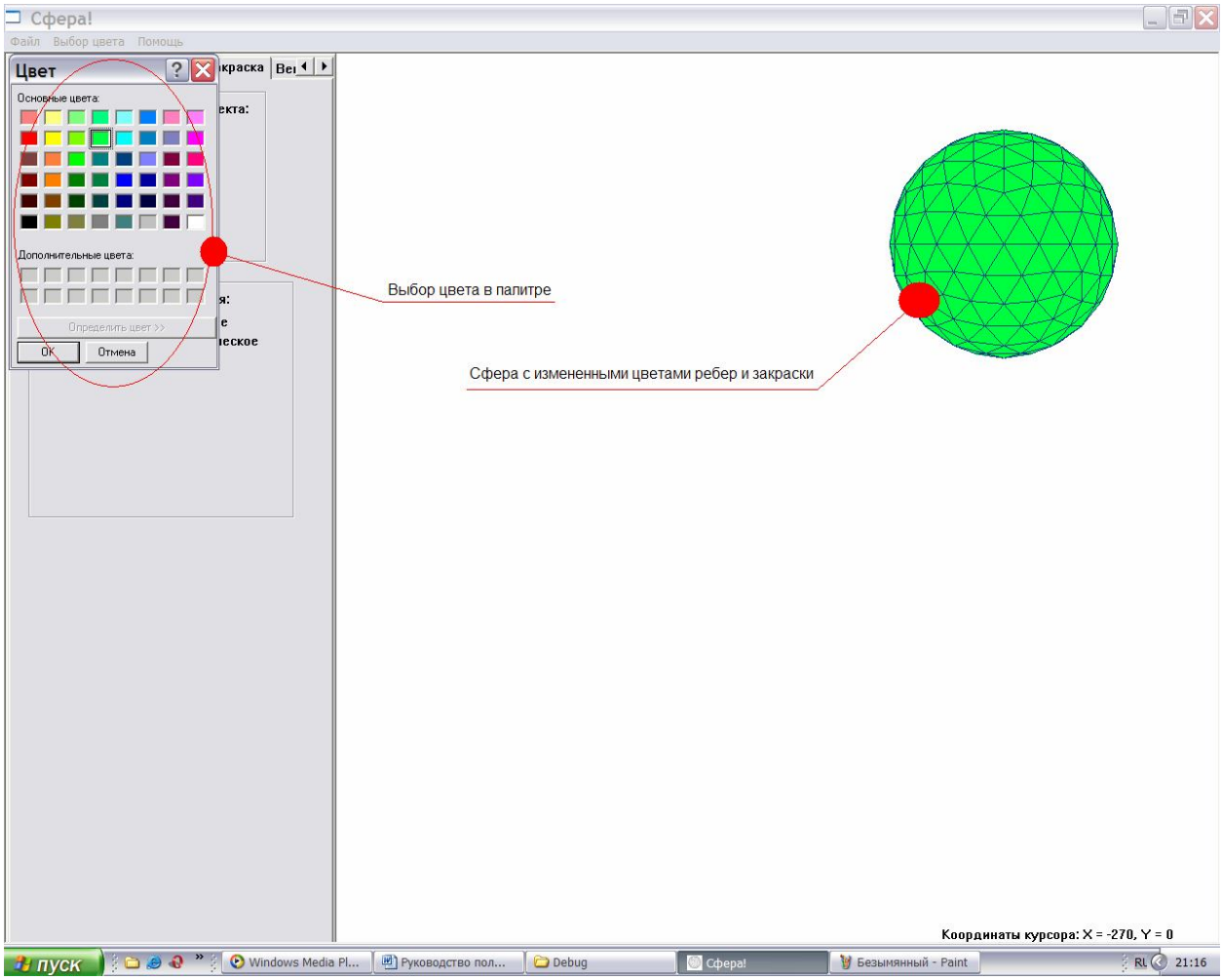


Рисунок 29. Сфера с цветами, выбранными пользователем.

На этой же вкладке можно устанавливать вид проецирования. Установим вид центрального одноточечного проецирования переключателем в блоке “Виды проецирования” как показано на рисунке 30.

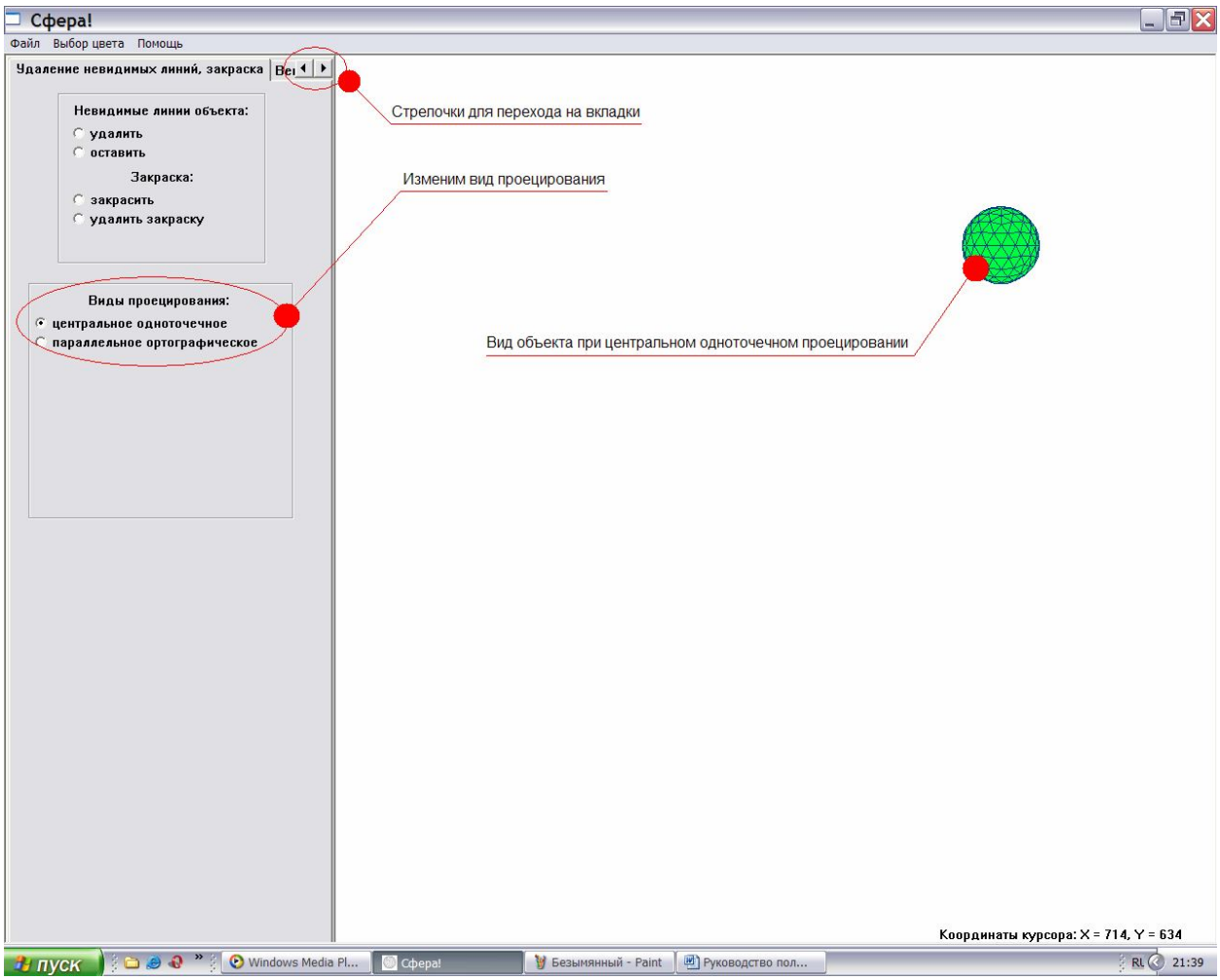


Рисунок 30. Отображение сферы при центральном одноточечном проецировании.

В данной программе реализовано вращение сферы вокруг произвольного вектора, параметры которого задаются на вкладке “Вектор”. Но прежде чем задавать параметры вектора, давайте изменим параметры отображения самой сферы. Вид проецирования – параллельный ортогональный. Далее переходим на вкладку каркасная модель (На рисунке вкладка “Каркасная модель” не видна. Для того чтобы она появилась необходимо нажать на стрелочки расположенные справа от названий вкладок. Данная операция необходима из-за достаточно длинных названий вкладок.) и устанавливаем координаты центра проекции объекта по X:450, и по Y:450. Это мы уже умеем делать. Остальные параметры оставляем без изменений как показано на рисунке 31.

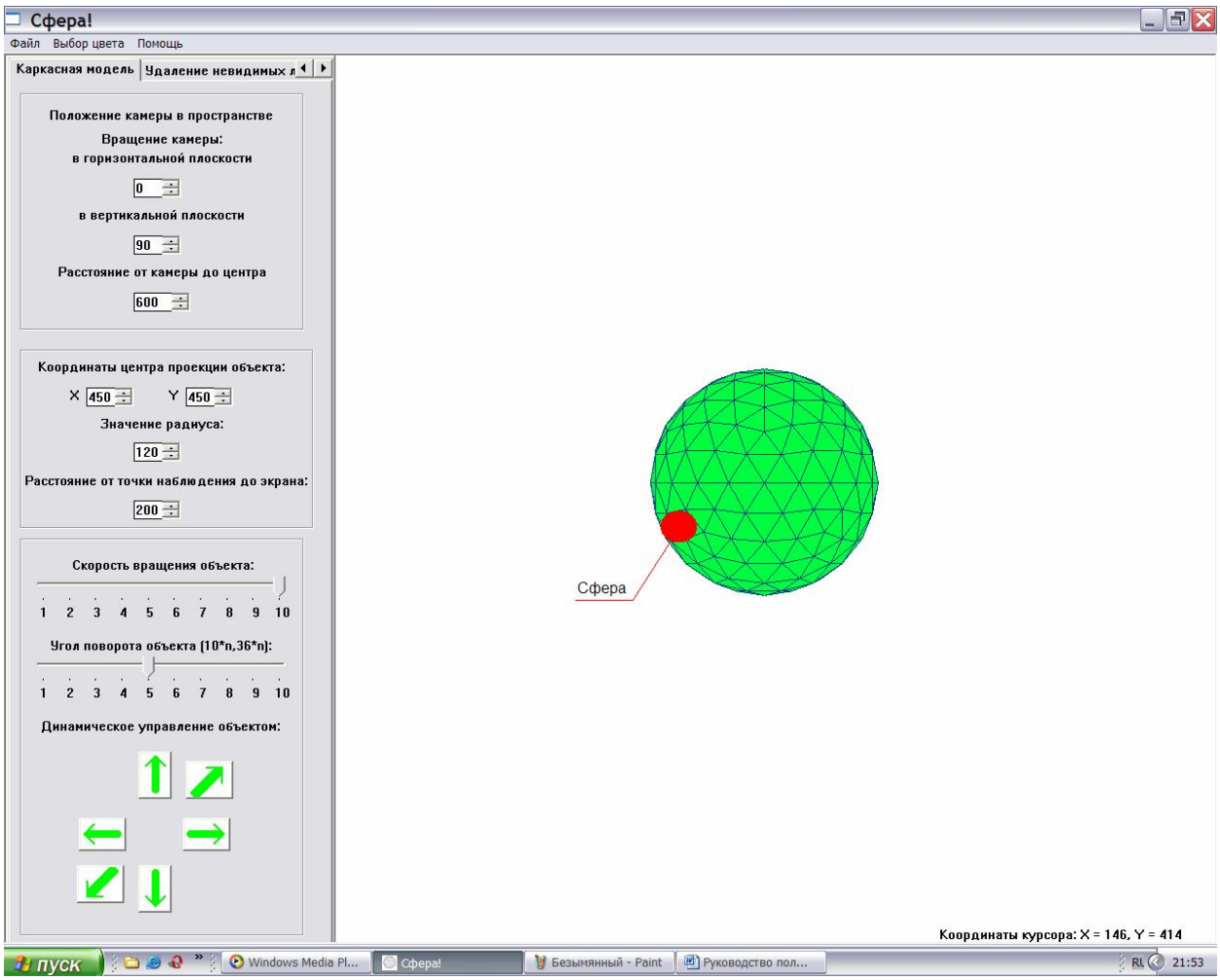


Рисунок 31. Результат установки параметров.

Теперь можно перейти на вкладку “Вектор”, воспользовавшись все теми же пресловутыми стрелочками справа от названий вкладок, рисунок 32.

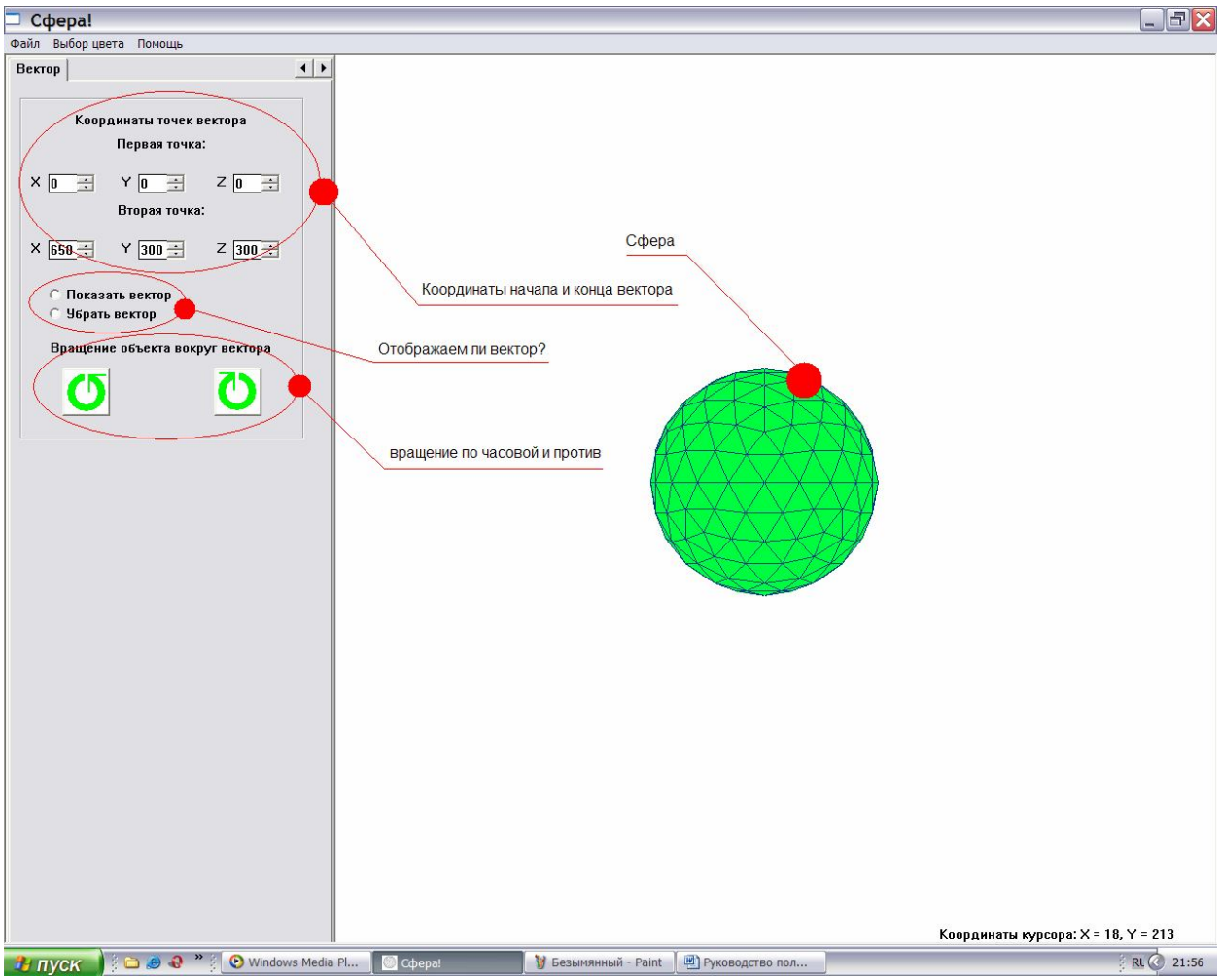


Рисунок 32. Элементы управления на вкладке “Вектор”.

На данном рисунке видны координаты концов вектора, кнопки поворота сферы вокруг вектора (скорость поворота и угол можно выбрать на вкладке “Каркасная модель” как мы это делали раньше), а также переключатели для его показа. Изменим эти параметры и посмотрим, что произойдет. Координаты первой точки X:0, Y:250, Z:300, Координаты второй точки оставим без изменений. Установим переключатель в положение “Показать вектор”. Установим цвет для данного вектора в пункте основного меню: Цвет->Выбор цвета->вектора (А заодно и цвет ребер сферы и ее закраску. Это делать мы уже умеем.). Результат данных операций представлен на рисунке 33.

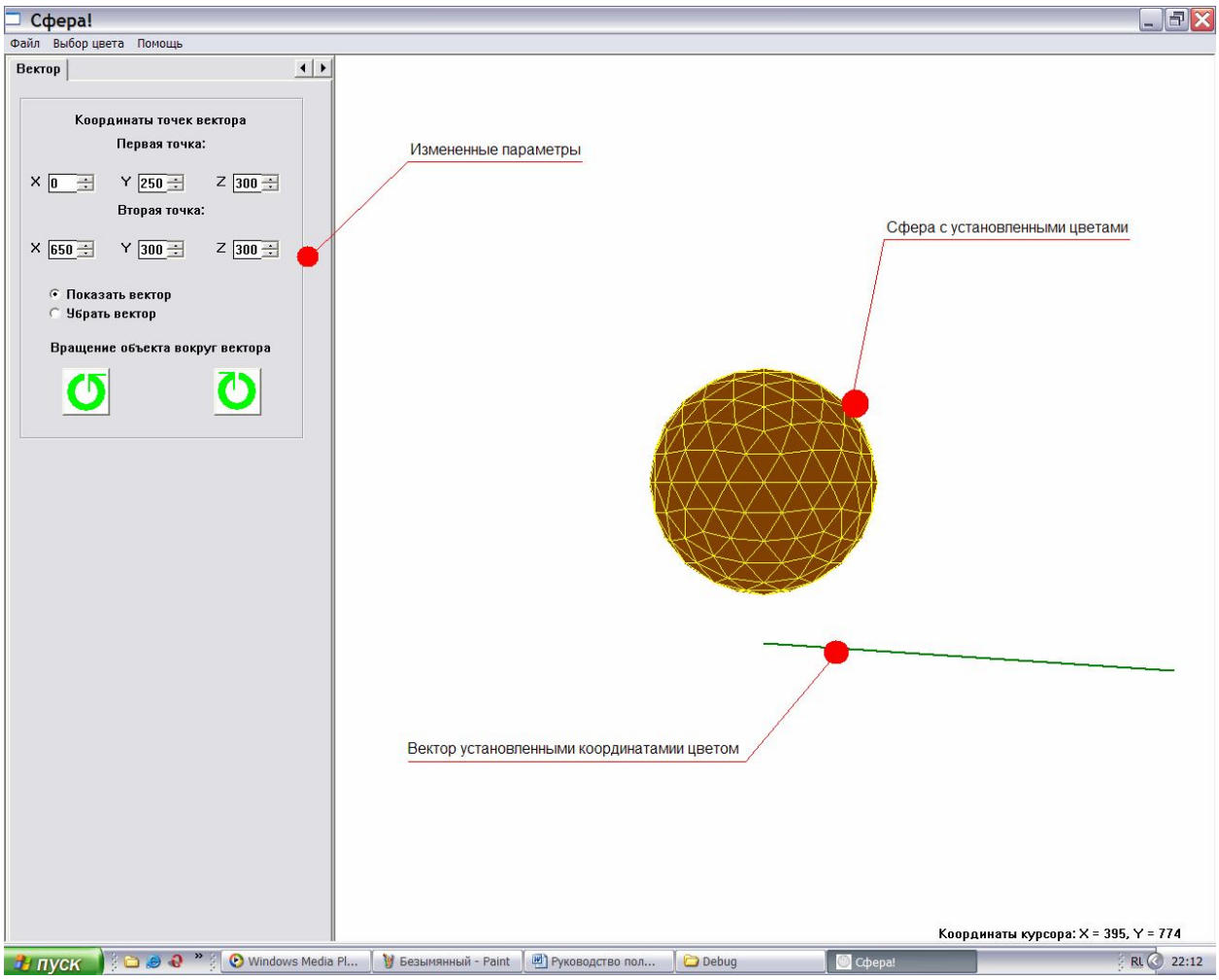


Рисунок 33. Сфера и установленный вектор.

Повернем сферу вокруг вектора, нажав на любую из двух кнопок как показано на рисунке 34.

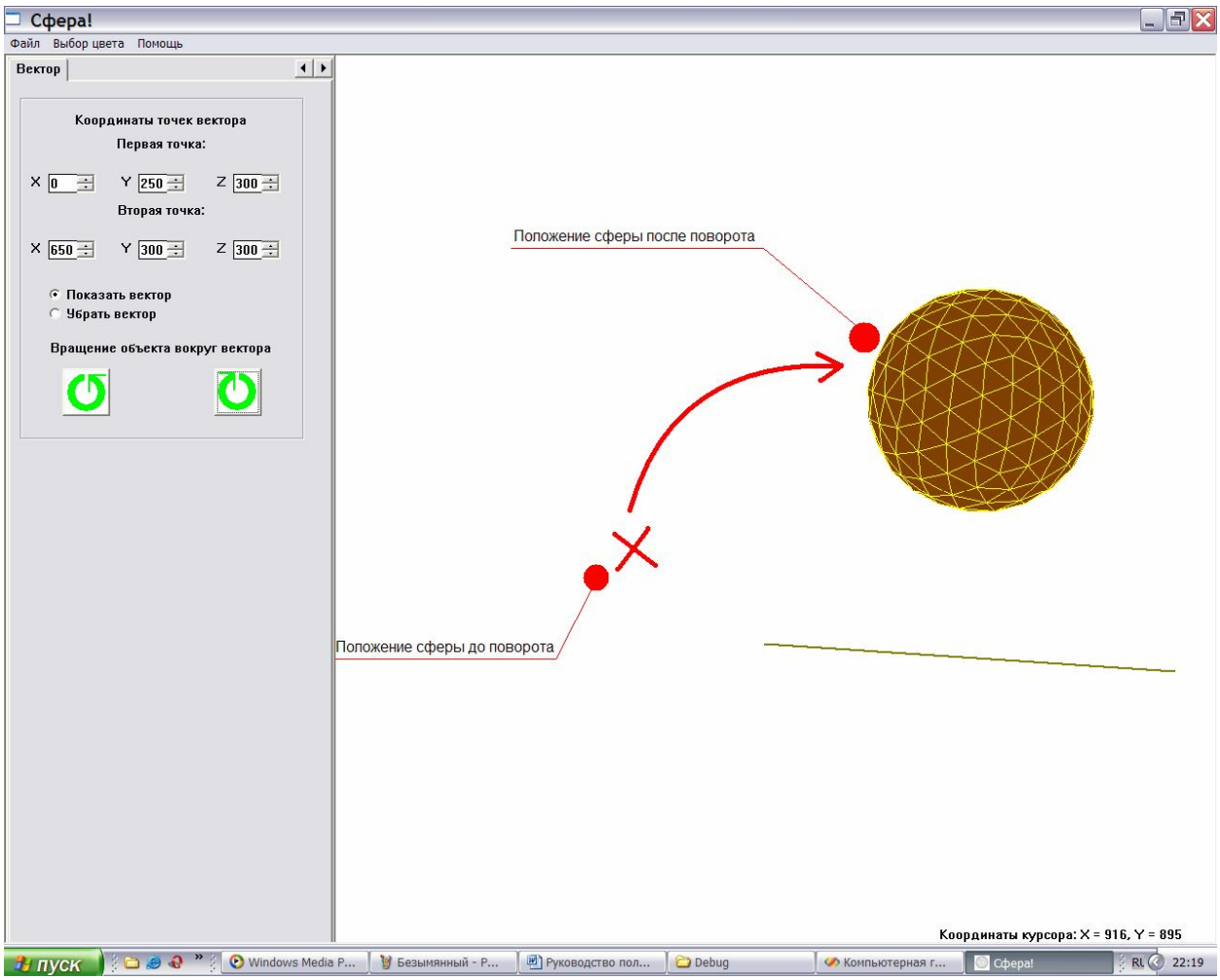


Рисунок 34. Поворот сферы вокруг вектора.

Теперь зададим координаты первой точки вектора: X:0, Y:0, Z:0, координаты второй точки оставим без изменений как показано на рисунке 35.

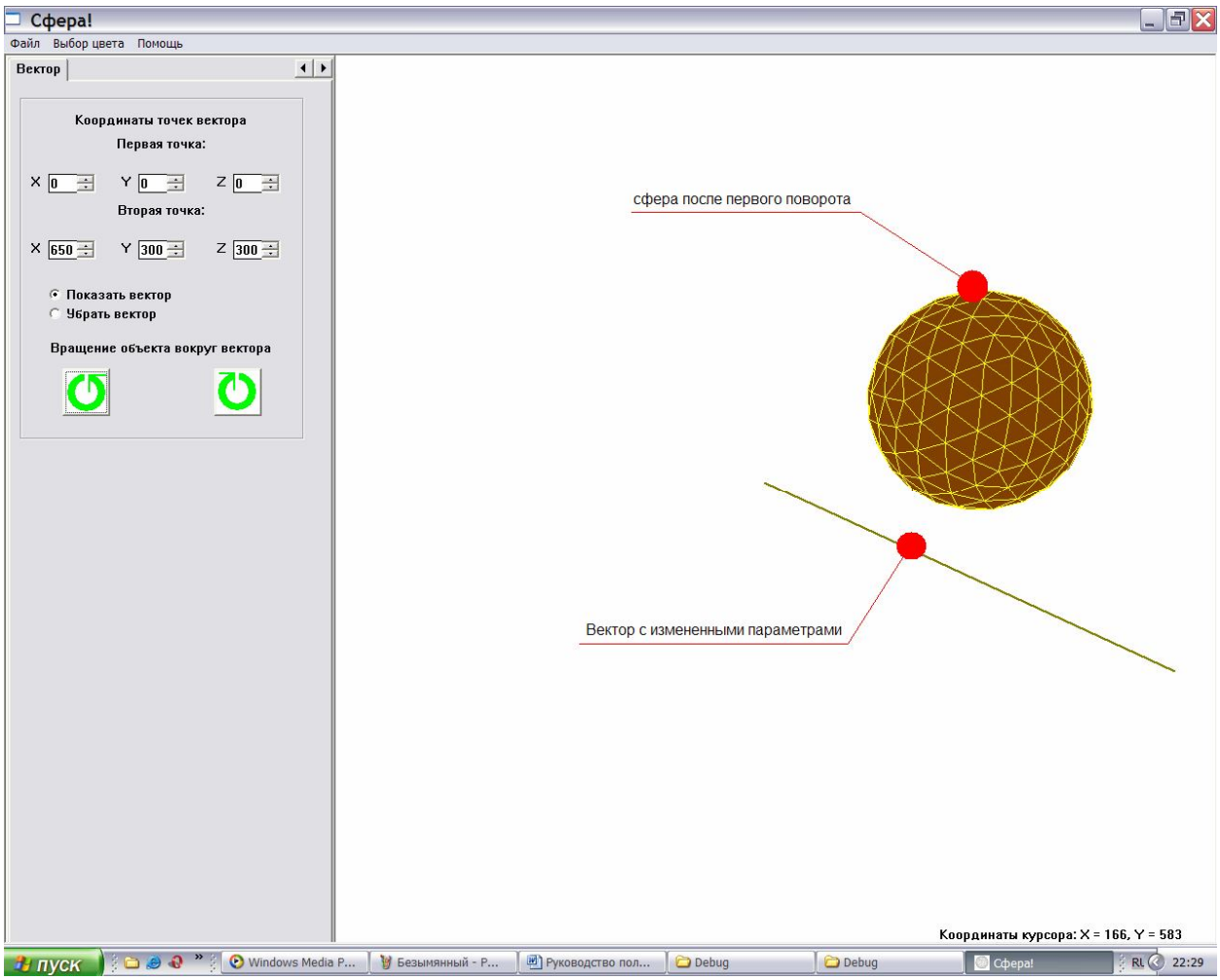


Рисунок 35. Сфера и вектор с измененными параметрами.

Повернем сферу вокруг вектора, нажав на любую из двух кнопок как показано на рисунке 36.

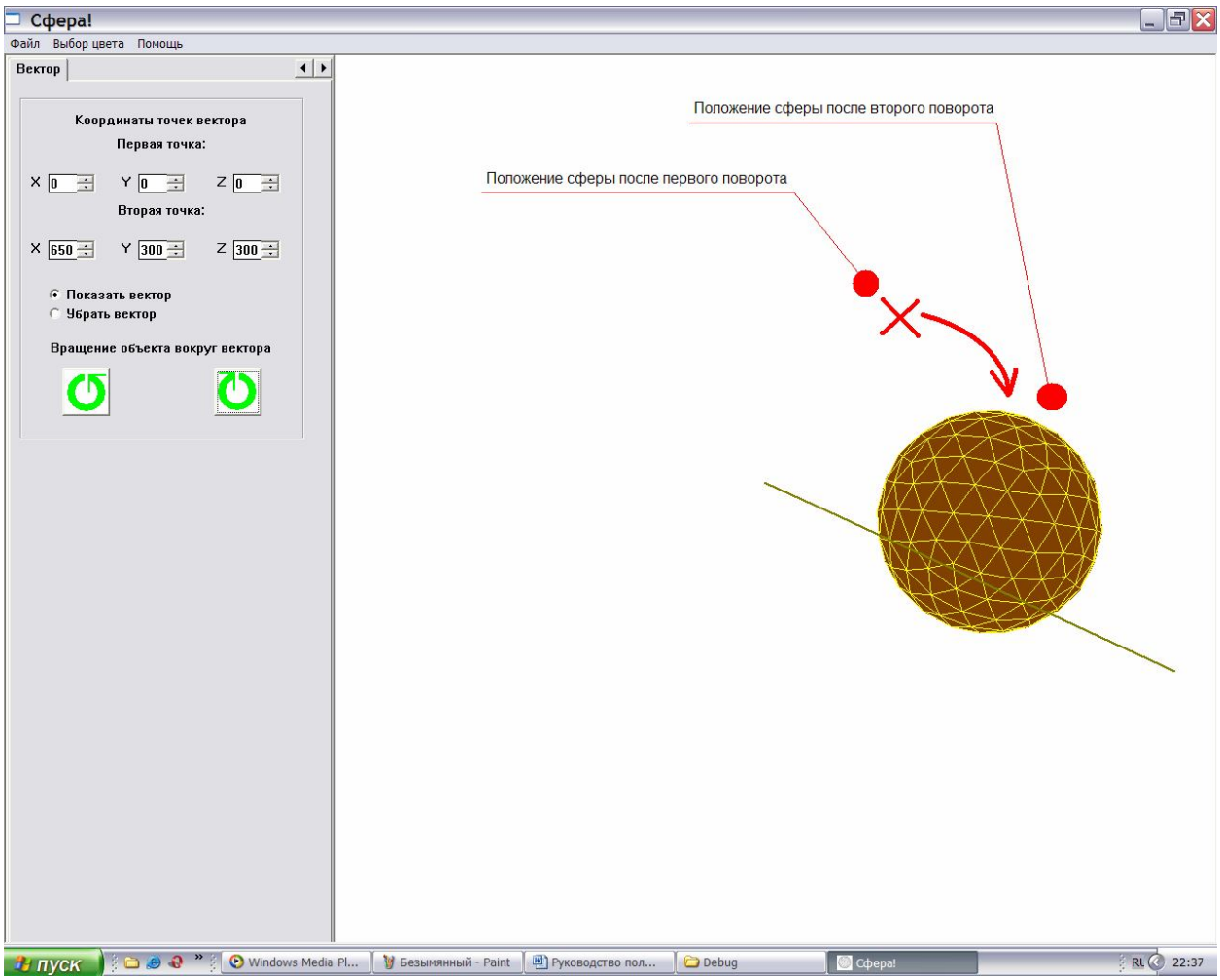


Рисунок 36. Поворот сферы вокруг вектора.

Настало время изменения параметров камеры. Оставим вектор на экране с теми координатами, которые мы установили, изменив лишь цвет вектора на черный (чтобы вектор лучше был виден на фоне сферы). Перейдем на вкладку “Удаление невидимых линий, закраска”. Установим вид проецирования – центральное одноточечное. Перейдем на вкладку “Каркасная модель”. Увеличим радиус сферы, установив его значение в 250. Результат данных операций показан на рисунке 37.

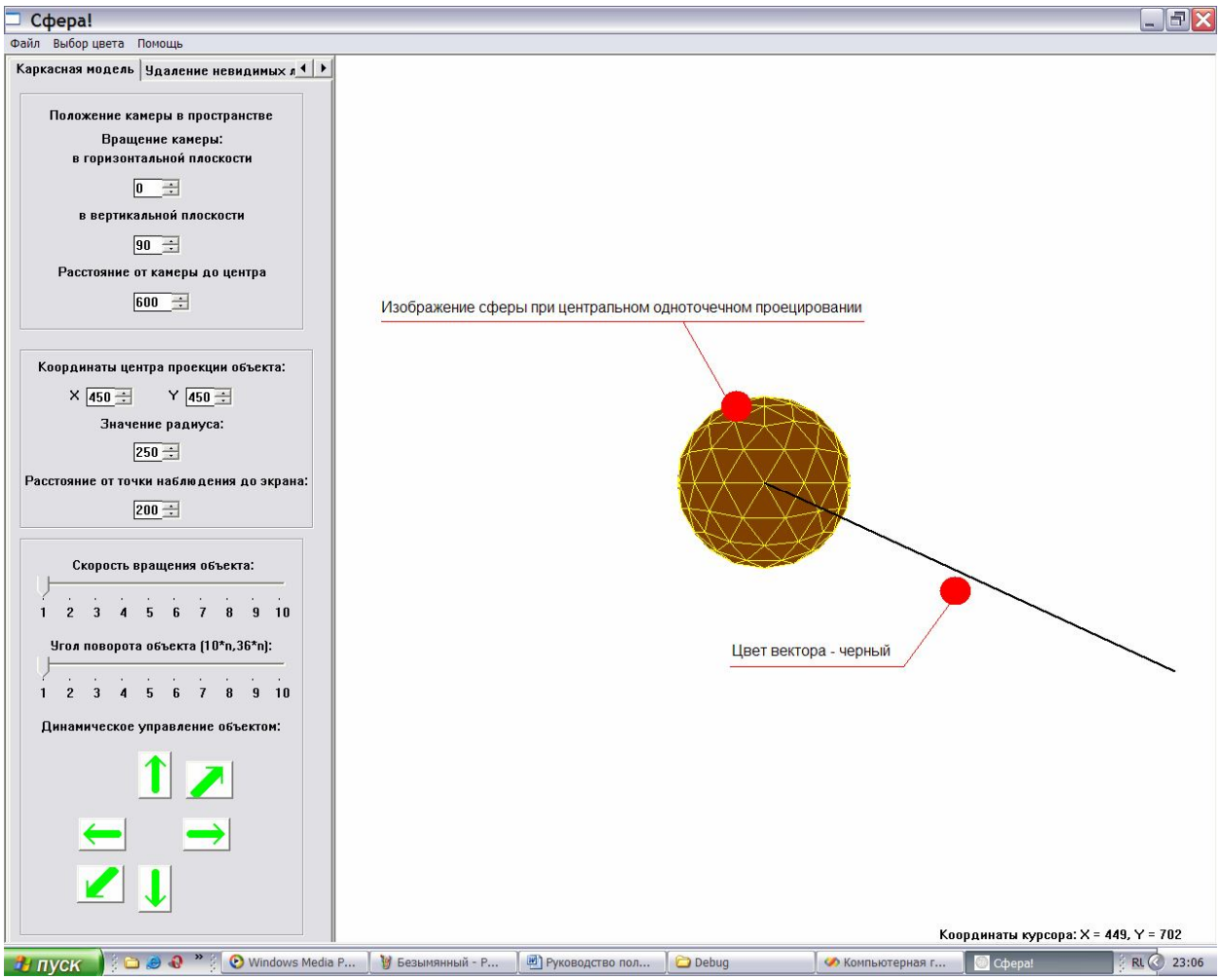


Рисунок 37. Сфера и вектор с измененными параметрами.

Изменим расстояние от точки наблюдения, установив его значение в 350. Теперь меняя угол камеры в горизонтальной (30 градусов, рисунок 38) и вертикальной(120 градусов, рисунок 39) плоскости, можно заметить, как изменяется изображение сферы и вектора.

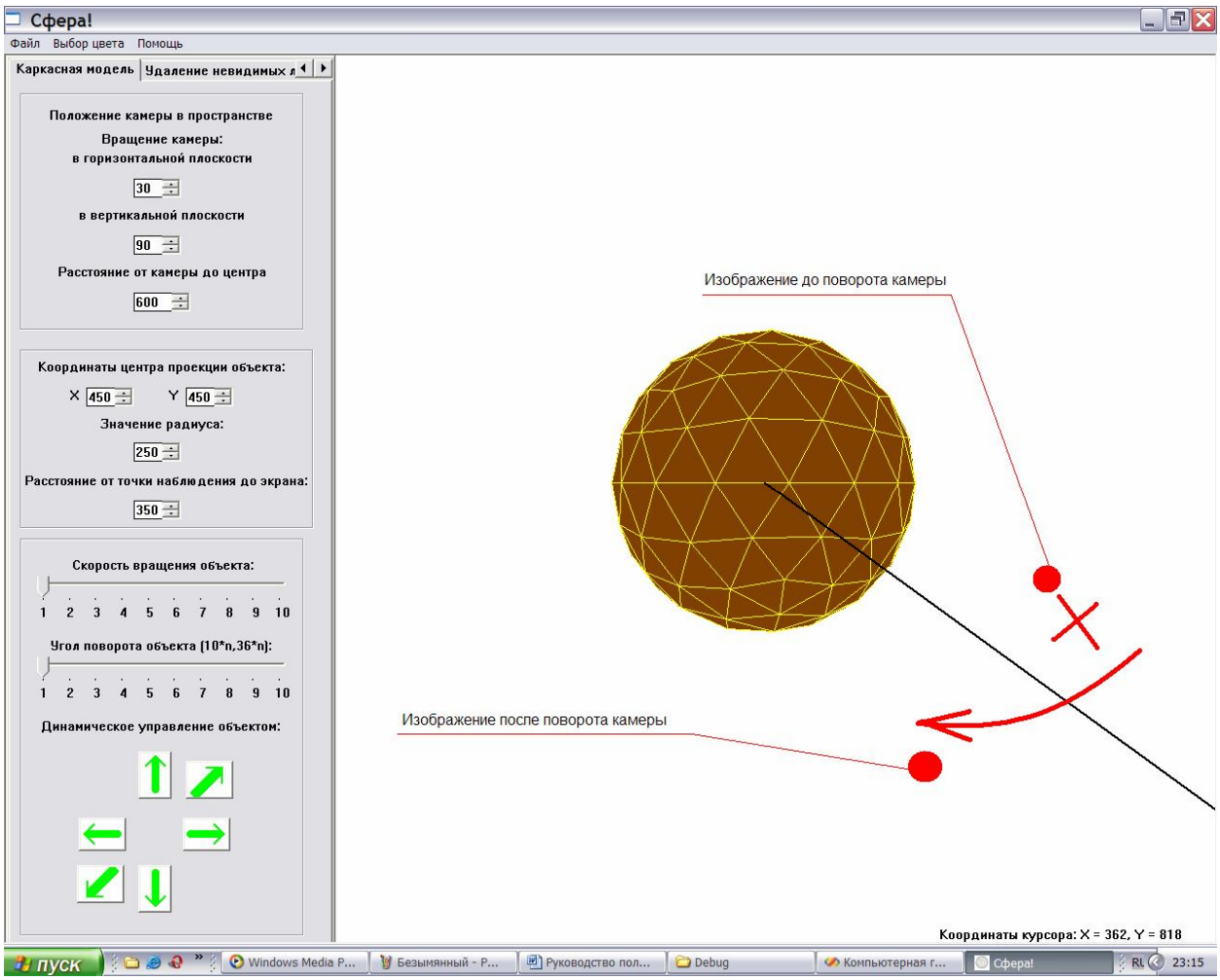


Рисунок 38. Изображение после поворота камеры в горизонтальной плоскости.

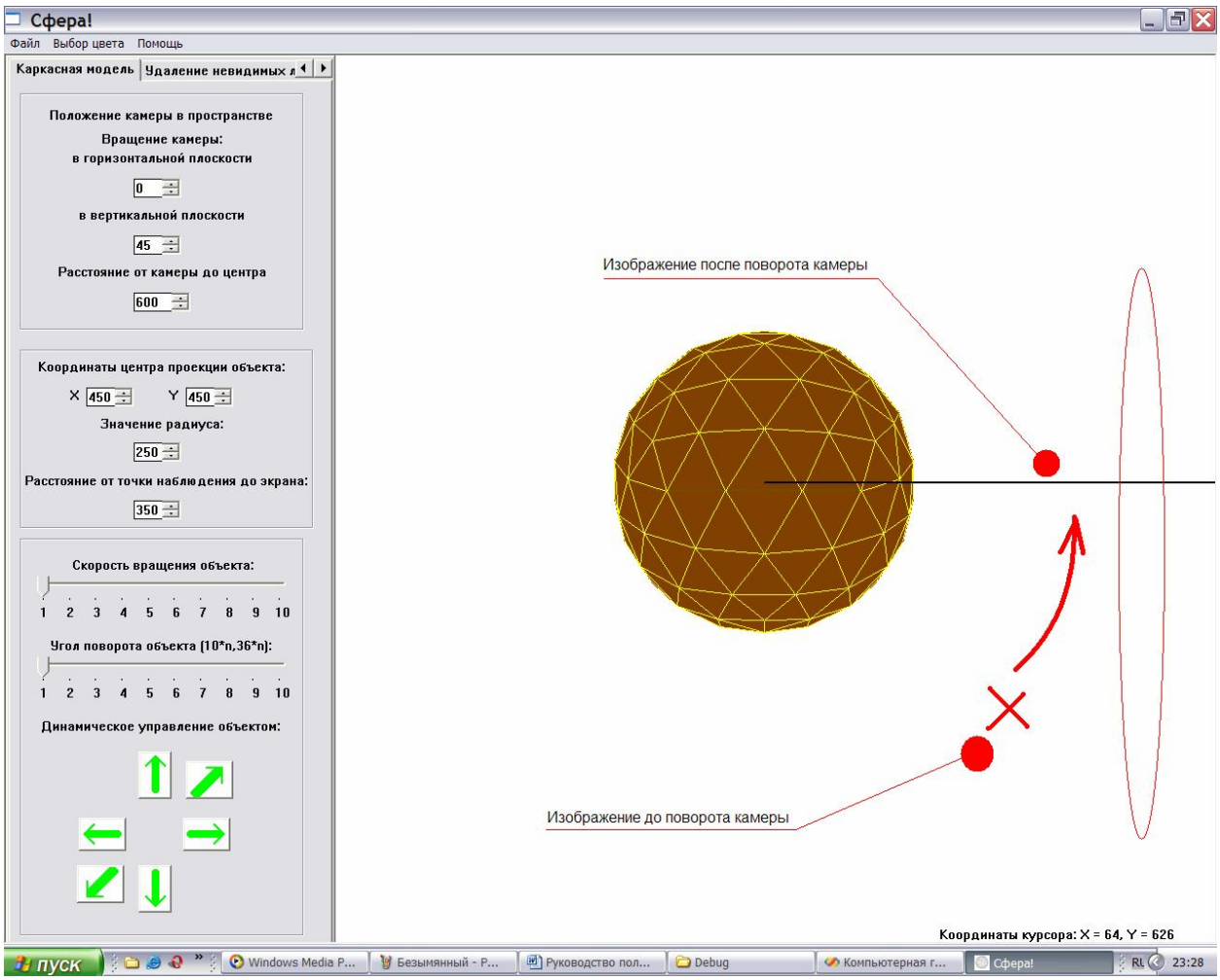


Рисунок 39. Изображение на экране после поворота камеры в вертикальной плоскости.

Нажав на пукт основного меню “О программе” вы вызовите диалоговое окно, выводящее возможности программы (см. рисунок 40).

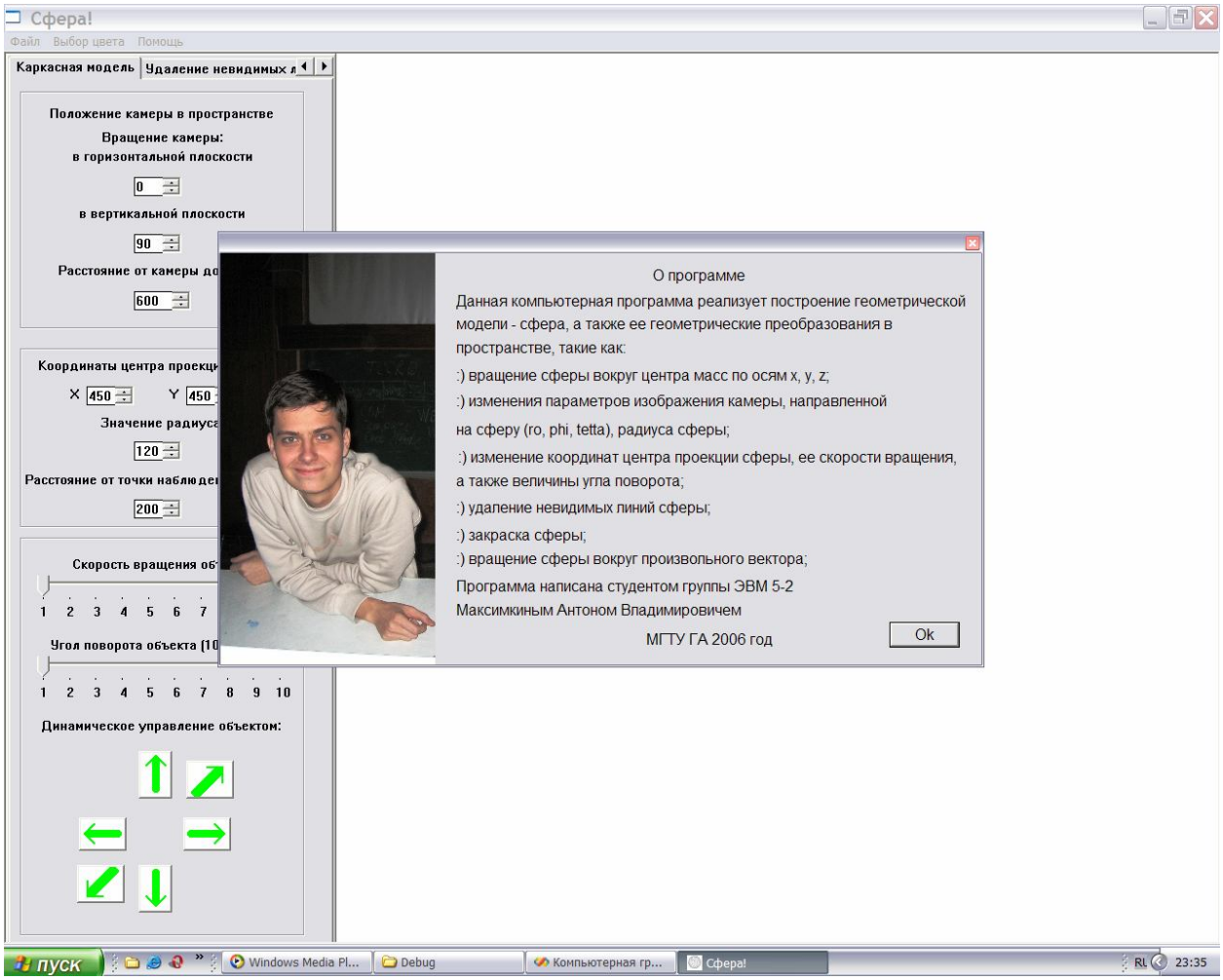


Рисунок 40. Диалоговая форма “О программе”.

Нажав на пункт основного меню Файл->Выход вы выйдите из программы (выйти из программы можно также нажав на кнопку с крестиком, расположенную в верхнем правом углу окна (здесь же кнопки свернуть и развернуть окно)) (см. рисунок 41).

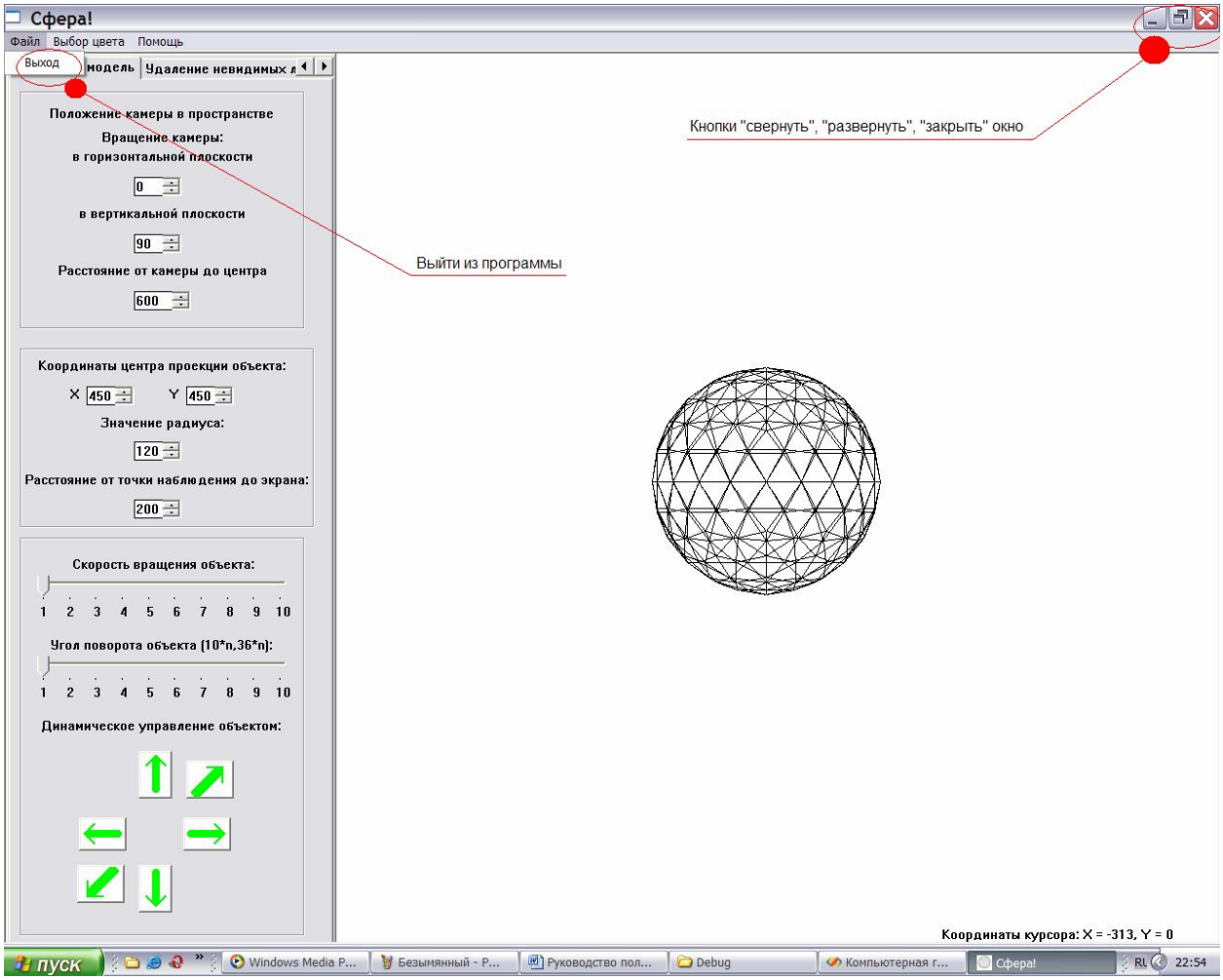


Рисунок 41. Элементы управления для выхода из программы.

Все вышеперечисленные операции вы можете проделывать в любой последовательности, в любом количестве и одновременно следить, как изменяется изображение сферы при изменении вами его параметров.

8. Приложения

Листинги программы представлены в приложении А. Виды экрана с графическими результатами представлены в разделе “Руководство пользователя”.

9. Литература:

1. Федотова Т. Н. Пособие к выполнению лабораторных работ по дисциплине “Компьютерная графика”. – М.: МГТУГА, 2002.
2. Федотова Т. Н. Конспект лекций по дисциплине “Компьютерная графика”.
3. Шикин А.В., Боресков А. В. Компьютерная графика. Полигональные модели. – М.: ДИАЛОГ – МИФИ, 2005.
4. Саймон Ричард. Microsoft Windows API. Справочник системного программиста. Второе издание, дополненное: Пер. с англ./Ричард Саймон – К.: ООО “ТИД “ДС”, 2004.
5. Верма Р. Д. Справочник по функциям Win32 API. – М.: Горячая линия – Телеком, 2002.
6. Гусев В. А., Мордкович А. Г. Математика: Справ. Материалы: Кн. Для учащихся. – М.: Просвещение, 1988.
7. Румянцев П. В. Азбука программирования в Win32 API. – 4-е изд. – М.: Горячая линия – Телеком, 2004.
8. Финогенов К. Г. Win32. Основы программирования.- М.: ДИАЛОГ – МИФИ, 2002.
9. Подбельский В. В. Язык Си++: Учеб. пособие. – 5-е изд. – М.: Финансы и статистика, 2001.
10. Блинова Т. А., Пореев В. Н. Компьютерная графика/Под ред. В. Н. Порева – К.: Издательство Юниор, СПб.: КОРОНА принт, К.: Век+, 2006.