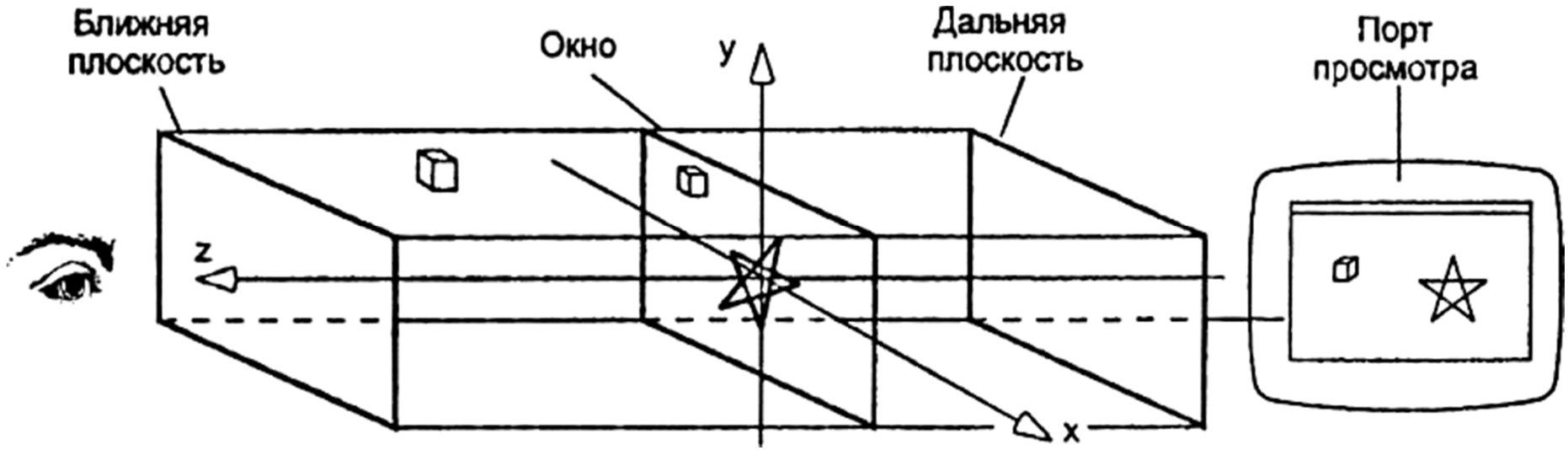
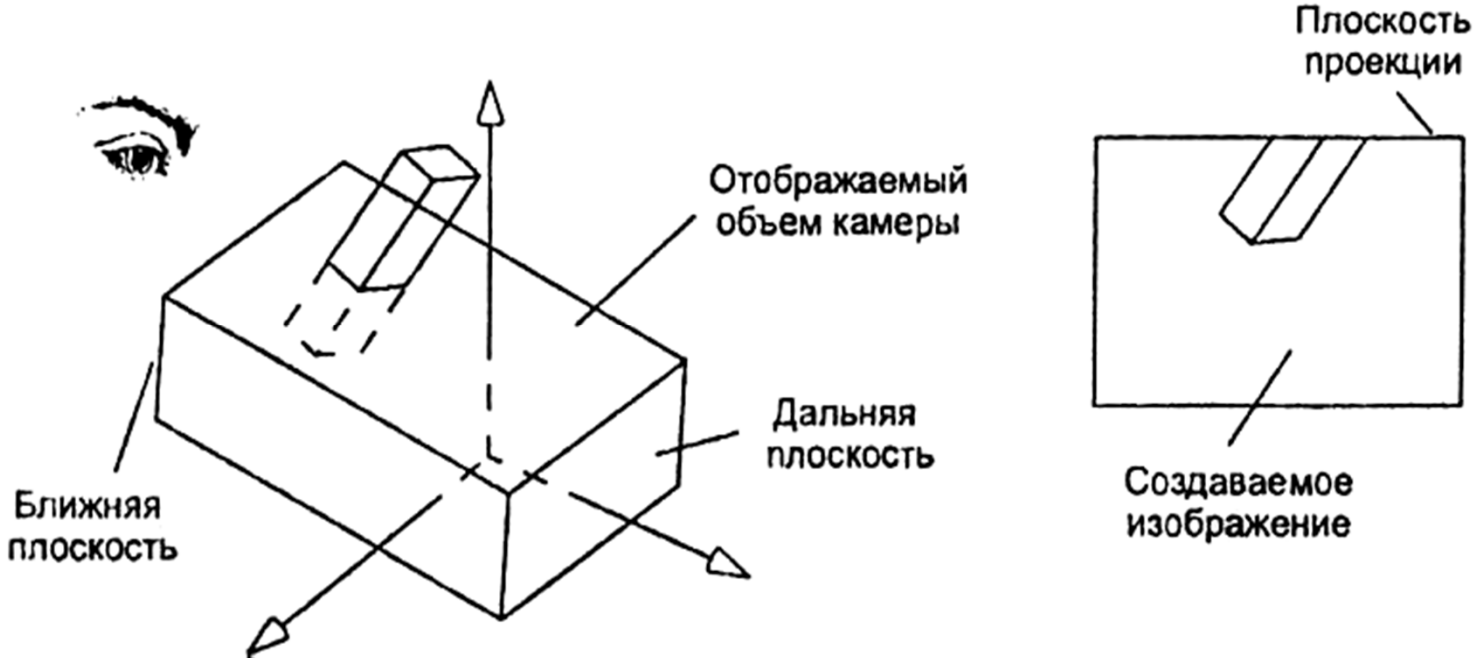
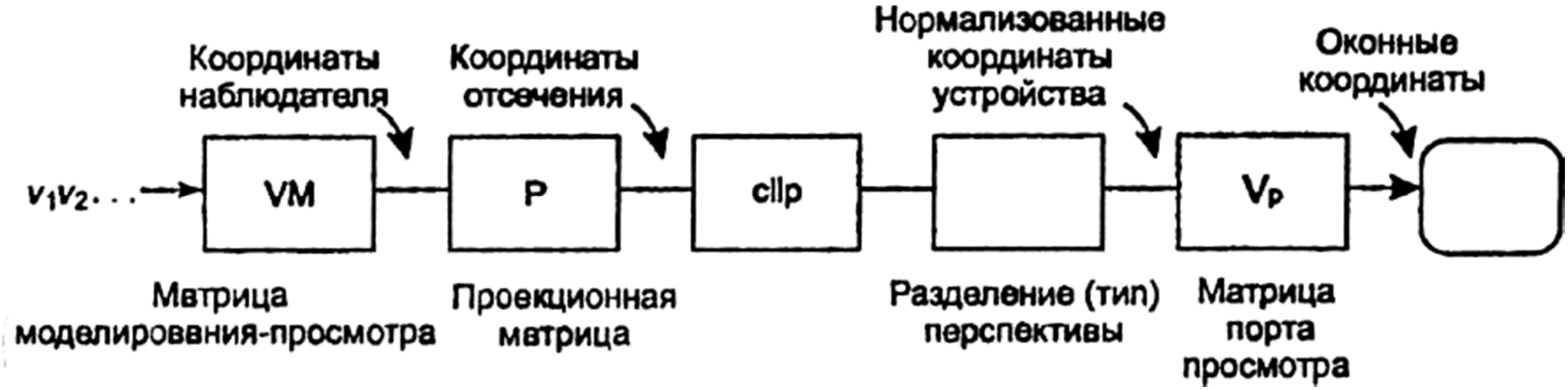
Простое визуальное отображение, используемое в OpenGL для двумерного рисования



Камера, создающая параллельные виды сцены



Конвейер OpenGL



Каждая вершина встречается со следующими тремя матрицами:

- матрица моделирования-вида;
- проекционная матрица;
- матрица порта просмотра.

Рисование трехмерных сцен в OpenGL

Матрица моделирования-вида компонует два действия:

1. Последовательность моделирующих преобразований, применяемых к объектам;
2. Преобразование, ориентирующее и позиционирующее камеру в пространстве.

Отсюда и ее название — моделирование и вид.

Матрица моделирования-вида является единственной в каждом действующем конвейере, проще представлять ее в виде произведения двух матриц: матрицы моделирования M и матрицы вида (просмотра) V.

Сначала применяется матрица моделирования, а затем матрица вида, так что матрица моделирования-вида фактически является произведением матриц VM.

Рисование трехмерных сцен в OpenGL

Проекционная матрица масштабирует и перемещает каждую вершину особым способом, так что все те вершины, которые располагаются внутри отображаемого объема, будут располагаться внутри стандартного куба, расположенного в промежутках от -1 до 1 по каждому измерению.

Проекционная матрица эффективно преобразует отображаемый объем в куб с центром в начале координат, что является наиболее удобным геометрическим телом, границами которого производится отсечение объектов.

Кроме того, проекционная матрица инвертирует ось *z* в том смысле, что возрастающие значения *z* теперь означают большее удаление (глубину) точки от наблюдателя.

Рисование трехмерных сцен в OpenGL

Далее выполняется отсечение, которое отбрасывает ту часть параллелепипеда, которая лежит за пределами стандартного куба.

Матрица порта просмотра отображает «выжившую» часть параллелепипеда в «трехмерный порт просмотра». Эта матрица отображает стандартный куб в параллелепипед такой формы, величины которой x и y соответствуют размерам порта просмотра (в экранных координатах), а его z-компонент изменяется от 0 до 1 и содержит в себе глубину точки (расстояние от точки до глаза камеры).

Рисование трехмерных сцен в OpenGL

Таким образом, каждая точка V (являющаяся обычно вершиной полигона) проходит через следующие этапы:

1. Точка *V* расширяется до четверки однородных координат путем добавления 1.
2. На эту четверку умножается матрица моделирования-вида, в результате чего получается четверка, задающая положение точки в координатах наблюдателя.
3. Далее на эту точку умножается проекционная матрица, в результате чего получается четверка в координатах отсечения.
4. Ребро, содержащее спроецированную точку в качестве концевой, отсекается.

Рисование трехмерных сцен в OpenGL

1. Выполняется перспективное деление, возвращающее упорядоченную триаду.
2. Преобразование в порт просмотра умножает на эту тройку матрицу; результат умножения (sx, sy, dz) используется для рисования и для вычислений глубины. Точка (sx, sy) отображается в экранных координатах; величина dz является мерой расстояния (глубины) исходной точки от глаза камеры.

Установка камеры в OpenGL (для случая параллельной проекции)

Функция glOrtho(left, right, bottom, top, near, far); устанавливает в качестве отображаемого объема параллелепипед, располагающийся от left до right по оси x, от bottom до top по оси y и от -near до -far по оси *z*.

Так как по умолчанию камера расположена в начале координат и смотрит вдоль отрицательных значений оси *z*, использование для near значения 2 означает помещение ближней плоскости в $z = -2$, то есть на расстояние в две единицы перед глазом. Аналогично, использование 20 для far помещает дальнюю плоскость в 20 единицах перед глазом.

Установка камеры в OpenGL (для случая параллельной проекции)

Установка проекционной матрицы осуществляется с помощью следующего кода:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left, right, bottom, top, near, far);
```

Установка камеры в OpenGL (для случая параллельной проекции)

В OpenGL предлагается функция, которая упрощает установку основной камеры:

```
gluLookAt (eye_x, eye_y, eye_z, look_x, look_y, look_z, up_x, up_y, up_z);
```

eye — положение наблюдателя (камеры); точка, на которую направлен взгляд — look; up — приблизительное направление вверх.

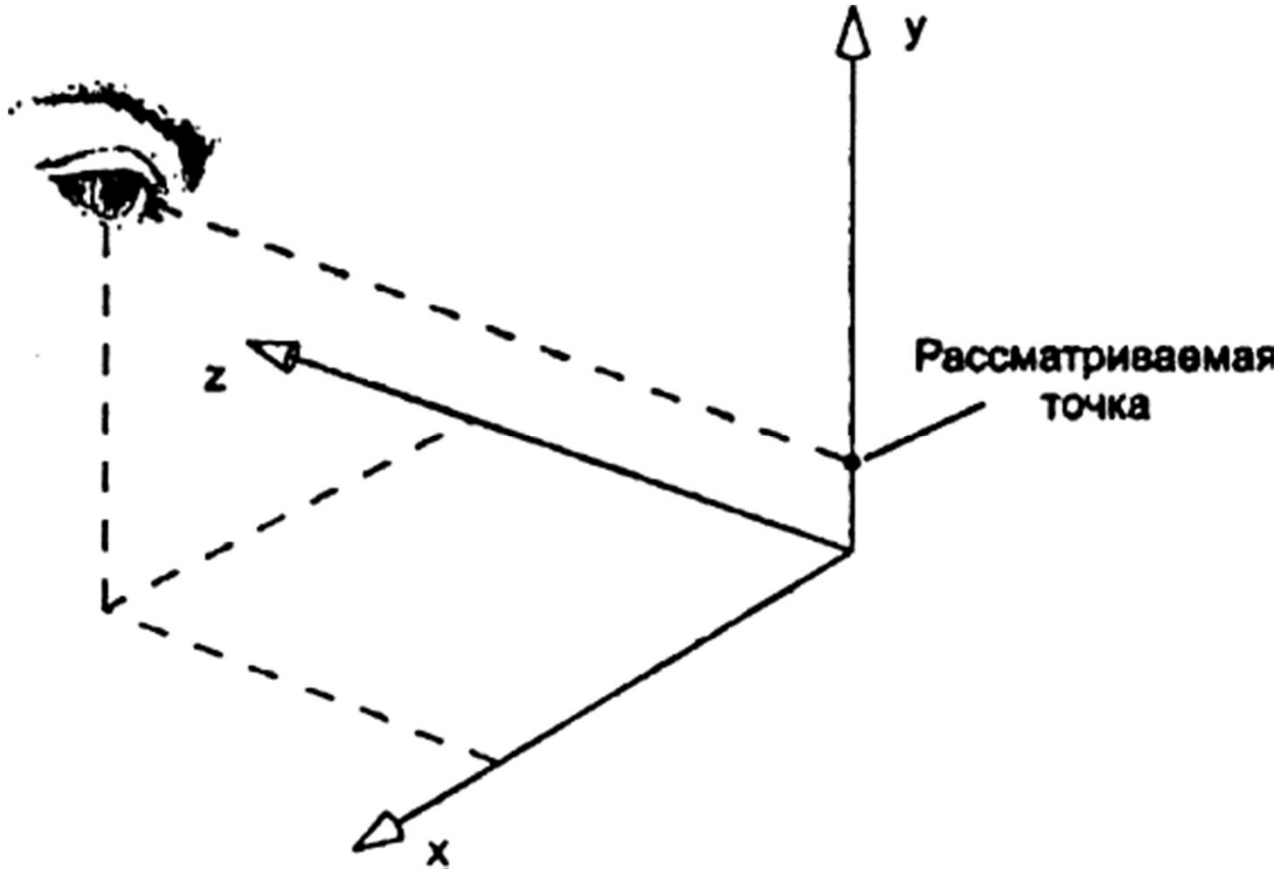
Установка камеры в OpenGL (для случая параллельной проекции)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eye_x, eye_y, eye_z, look_x, look_y,
look_z, up_x, up_y, up_z);
```

Установка камеры в OpenGL (для случая параллельной проекции)

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-3.2, 3.2, -2.4, 2.4, 1, 50);
glMatrixMode(GL_MODELVIEW);
gluLookAt(4, 4, 4, 0, 1, 0, 0, 1, 0);
```

Установка камеры в OpenGL (для случая параллельной проекции)



Рисование элементарных форм, поддерживаемых OpenGL

- куб: glutWireCube(GLdouble size); каждая сторона имеет длину size;
- сфера: glutWireSphere(GLdouble radius, GLint nSlices, GLint nStacks);
- тор: glutWireTorus(GLdouble inRad, GLdouble outRad, GLint nSlices, GLint nStacks);
- чайник: glutWireTeapot(GLdouble size).

Рисование элементарных форм, поддерживаемых OpenGL

Для визуализации четырех из Платоновых тел (пятым является куб, который уже был представлен) используются следующие функции:

- тетраэдр: glutWireTetrahedron();
- октаэдр: glutWireOctahedron();
- додекаэдр: glutWireDodecahedron();
- икосаэдр: glutWireIcosahedron().

Рисование элементарных форм, поддерживаемых OpenGL

glutWireCone(GLdouble baseRad, GLdouble height. GLint nSlices. GLint nStacks);

Фигуры аппроксимируются полигональными гранями, поэтому путем изменения параметров nSlices и nStacks можно устанавливать, сколько граней будет использовано в аппроксимации. Параметр nSlices — это число «долек» вокруг оси z, а nStacks — число «ломтиков» вдоль оси z, как если бы данная форма была стопкой из множества nStacks дисков.

Рисование элементарных форм, поддерживаемых OpenGL

```
GLUQuadricObj * qobj =gluNewQuadric(); // создаем квадратичный объект
gluQuadricDrawStyle(qobj, GLU_LINE); // устанавливаем стиль каркасной модели
gluCylinder(qobj, baseRad, topRad, height, nSlices, nStacks); // рисуем цилиндр.
gluDeleteQuadric(qObj); // Освобождается память, которую занимал квадратичный объект
```

Рисование элементарных форм, поддерживаемых OpenGL

```
GLUQuadricObj *pObj; // Создается и инициализируется
pObj = gluNewQuadric(); // квадратичная поверхность
…
// Задаются параметры визуализации квадратичных поверхностей и рисуются поверхности
…
gluDeleteQuadric(pObj); // Освобождается память, которую занимала квадратичная поверхность
```

Стиль рисования квадратичной поверхности

```
void gluQuadricDrawStyle(GLUquadricObj *obj, GLenum drawStyle);
```

Константа	Описание
GLU_FILL	Квадратичные объекты рисуются как сплошные
GLU_LINE	Квадратичные объекты рисуются как каркасные
GLU_POINT	Квадратичные объекты рисуются как набор точек-вершин
GLU_SILHOUETTE	Похоже на каркасное изображение, только смежные грани многоугольников не рисуются

Нормали

```
void gluQuadricNormals(GLUquadricObj *pbj, GLenum normals);
```

Квадратичные поверхности могут рисоваться без нормалей (GLU_NONE), с гладкими нормалями (GLU_SMOOTH) или с плоскими нормалями (GLU_FLAT).

```
void gluQuadricOrientation(GLUquadricObj *obj, GLenum orientation);
```

Значением параметра orientation может быть GLU_OUTSIDE либо GLU_INSIDE.

Текстура

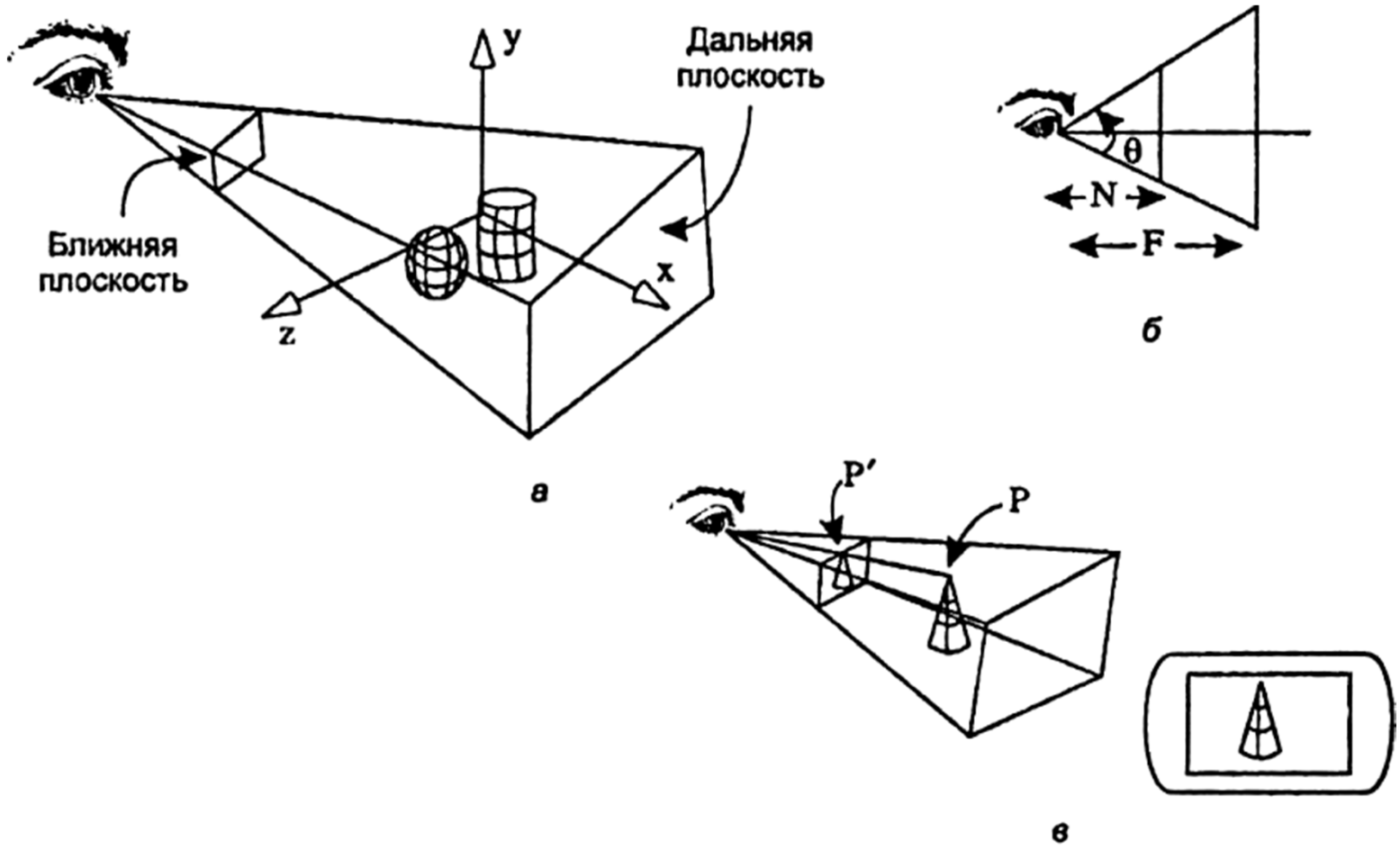
```
void gluQuadricTexture (GLUquadricObj *obj,GLenum textureCoords);
```

Параметр textureCoords может иметь значение GL_TRUE либо GL_FALSE. При генерации текстурных координат для квадратичных поверхностей текстуры равномерно оборачиваются вокруг сфер и цилиндров, при наложении текстуры на диск центр текстуры совмещается с центром диска, а края текстуры выравниваются по краям диска.

Рисование поверхностей второго порядка

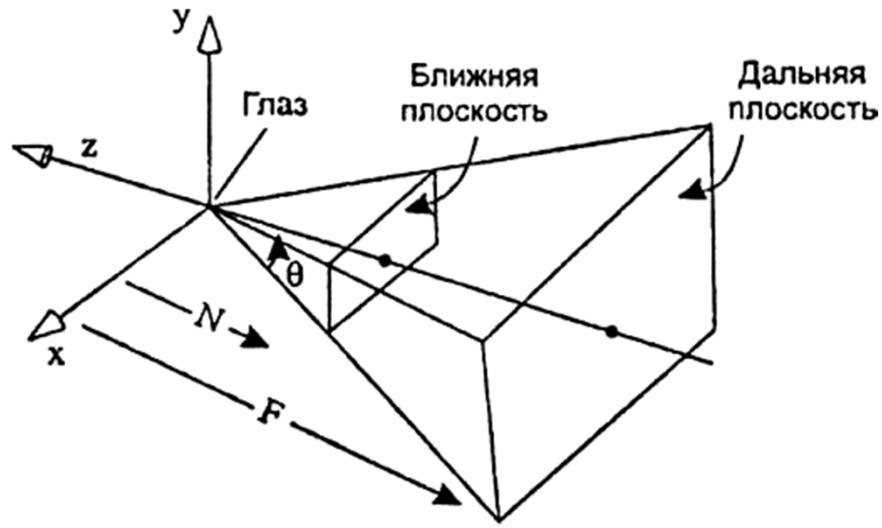
```
void gluSphere(GLUQuadricObj *obj,GLdouble radius,GLint slices, GLint stacks);
void gluCylinder(GLUquadricObj *obj,GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks);
void gluDisk(GLUquadricObj*obj,GLdouble innerRadius, GLdouble outerRadius, GLint slices,GLint loops);
```

Установка камеры в OpenGL (для случая перспективной проекции)



Установка камеры в OpenGL (для случая перспективной проекции)

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(viewAngle, aspectRatio, N, F);
```



Установка камеры в OpenGL (для случая перспективной проекции)

```
void glFrustum (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);
```

Объем видимости задается параметрами (left, bottom,-near) и (right, top,-near) определяющими координаты (x, y, z) левого нижнего и правого верхнего углов ближней отсекающей плоскости; near и far задают дистанцию от точки наблюдения до ближней и дальней отсекающих плоскостей (они всегда должны быть положительными).

Установка камеры в OpenGL (для случая перспективной проекции)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eye_x, eye_y, eye_z, look_x, look_y,
look_z, up_x, up_y, up_z);
```