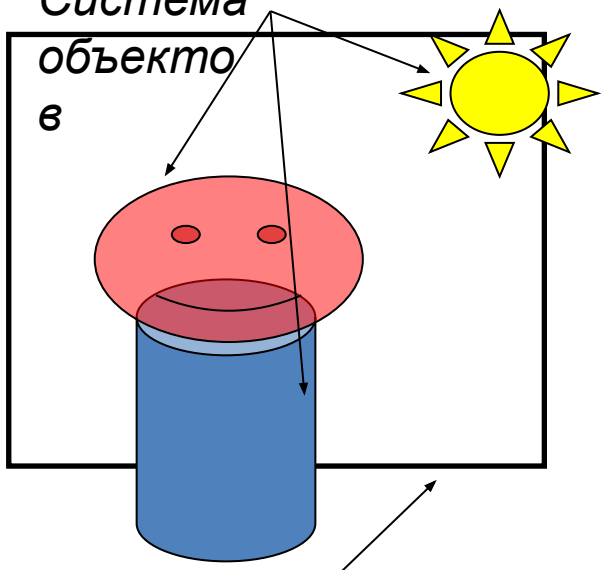
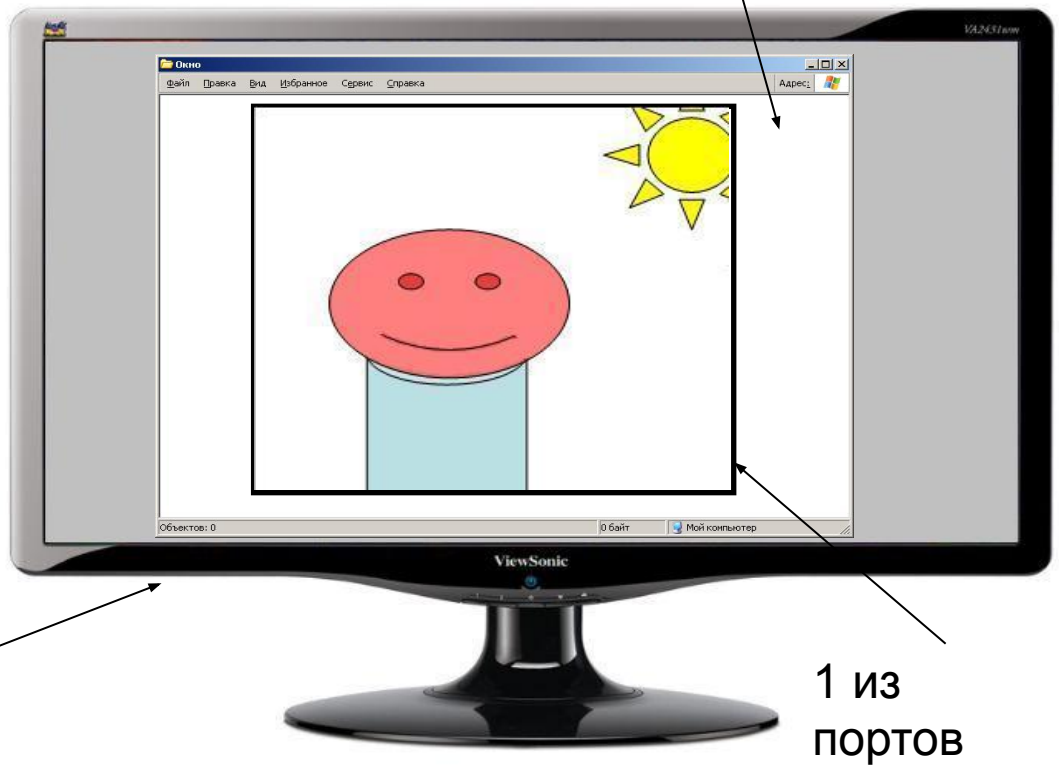
Интерактивная Компьютерная Графика

Часть 5-2

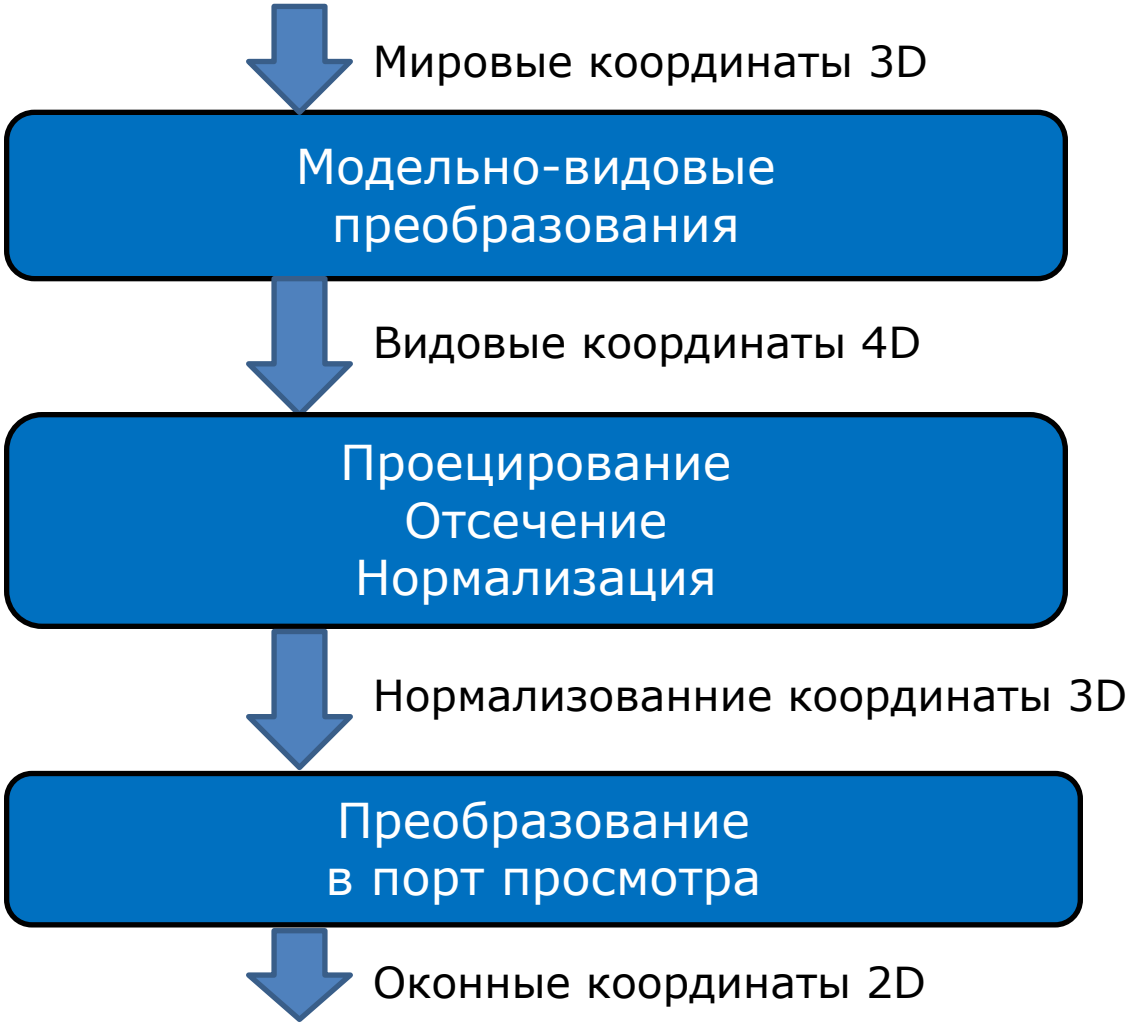
(системы координат)

Вход - выход


Система объектов
Мировое окно
Картинная плоскость


Экранное окно
Окно
Файл Правка Вид Избранное Сервис Справка
Адрес:
Объектов: 0
0 байт
Мой компьютер
ViewSonic
Экран
Дисплей
Монитор
1 из портов просмотра

Конвейер систем координат



Модельно-видовые преобразования:
сдвиг

Перемещение (сдвиг, перенос) на вектор $(dx, dy, dz)^T$

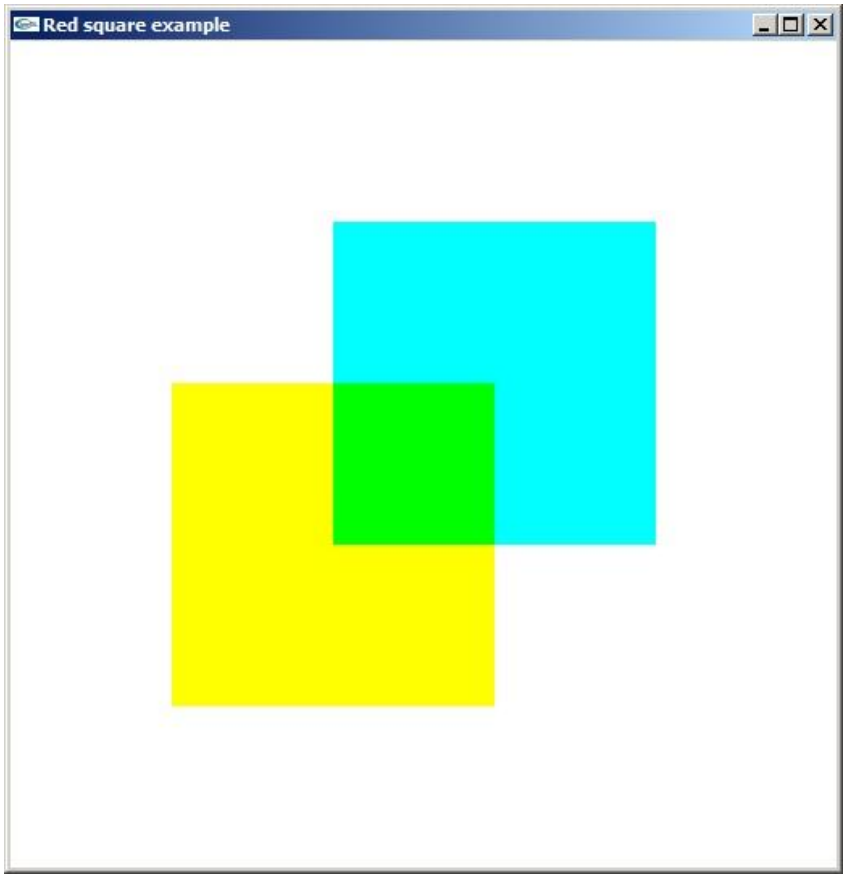
- x` = x + *dx*
- y` = y + *dy*
- z` = z + *dz*

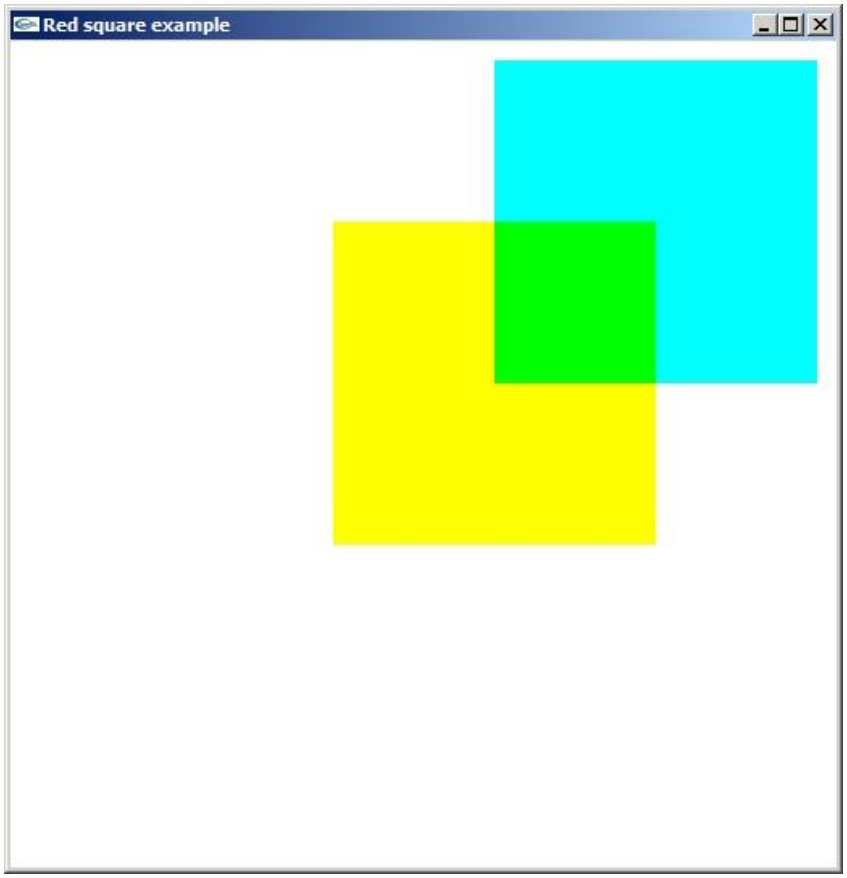
$$M_{trans} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
glTranslatef (dx, dy, dz)
```

Модельно-видовые преобразования:
сдвиг

```
glTranslatef ( 100, 100, 10 )
```





Модельно-видовые преобразования: масштабирование

Масштабирование (сжатие и растяжение) вдоль осей в (*sx, sy, sz*) раз

- x` = x * *sx*
- y` = y * *sy*
- z` = z * *sz*

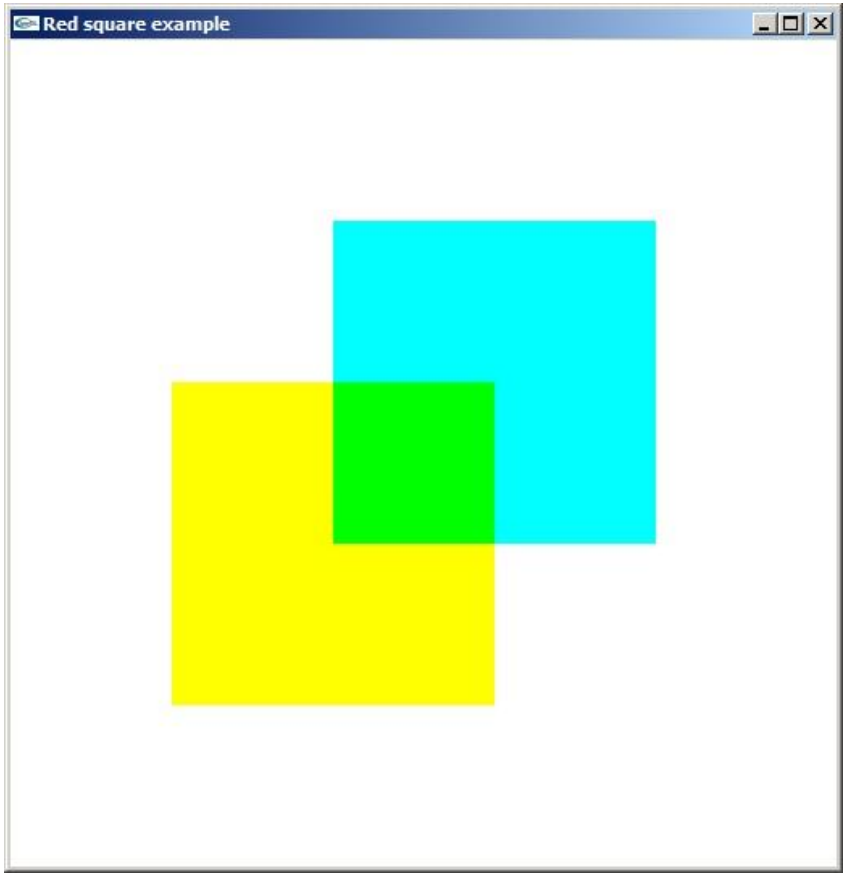
- Примечание: если один из параметров равен **0**, тогда преобразование не будет афинным

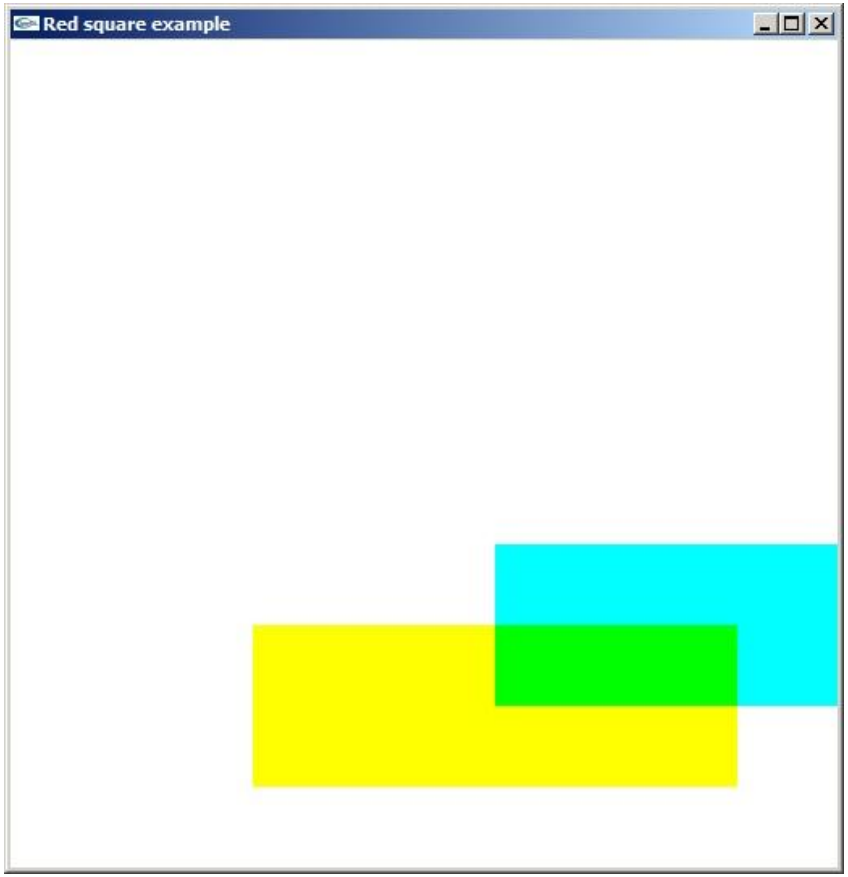
$$M_{scale} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

`glScalef (sx, sy, sz)`

Модельно-видовые преобразования:
масштабирование

```
glScalef ( 1.5, 0.5, 1 )
```





Модельно-видовые преобразования:
поворот

Поворот на угол angle против часовой стрелки относительно начала координат вдоль вектора (*rx, ry, rz*)

```
glRotatef (angle, rx, ry, rz)
```

$\text{glRotatef}\ (angle, 1, 0, 0)$ $\text{glRotatef}\ (angle, 0, 1, 0)$ $\text{glRotatef}\ (angle, 0, 0, 1)$

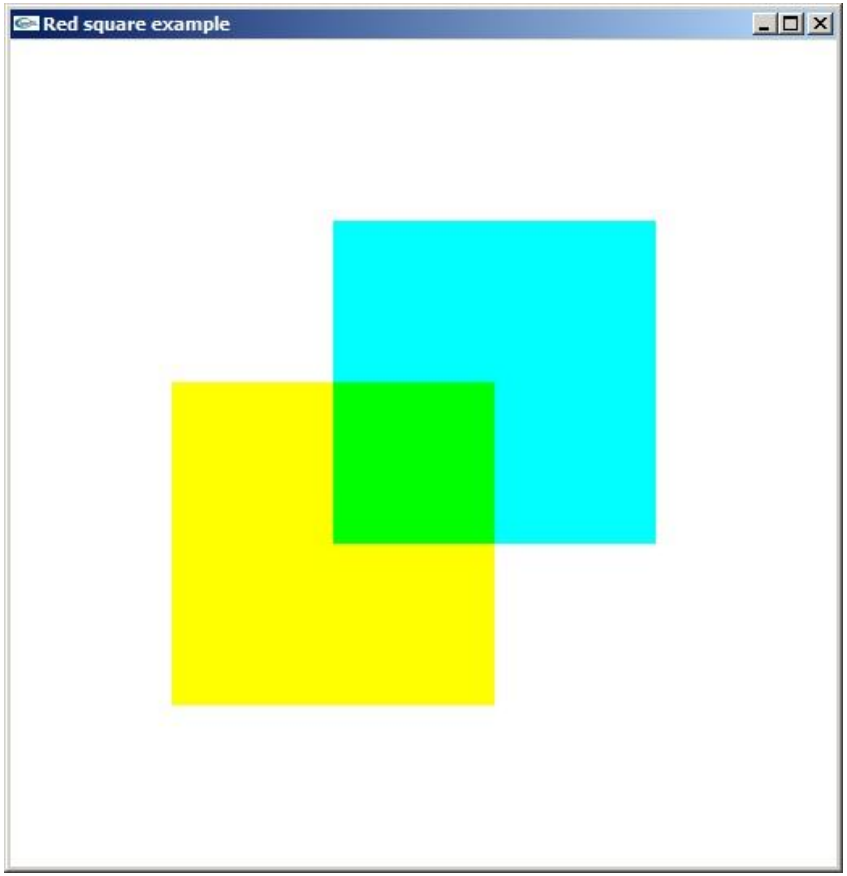
$$M_{OX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad M_{OY} = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad M_{OZ} = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

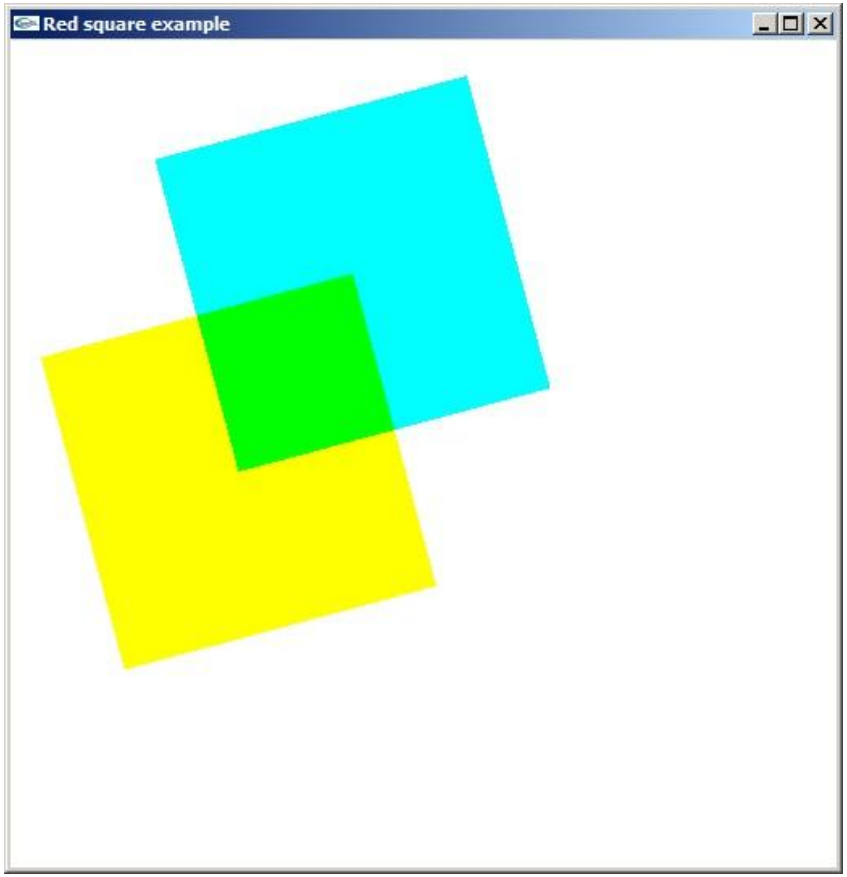
$$M_{rotate} = M_{OZ} * M_{OY} * M_{OX}$$

$$c = \cos(angle)$$
$$s = \sin(angle)$$

Модельно-видовые преобразования:
поворот

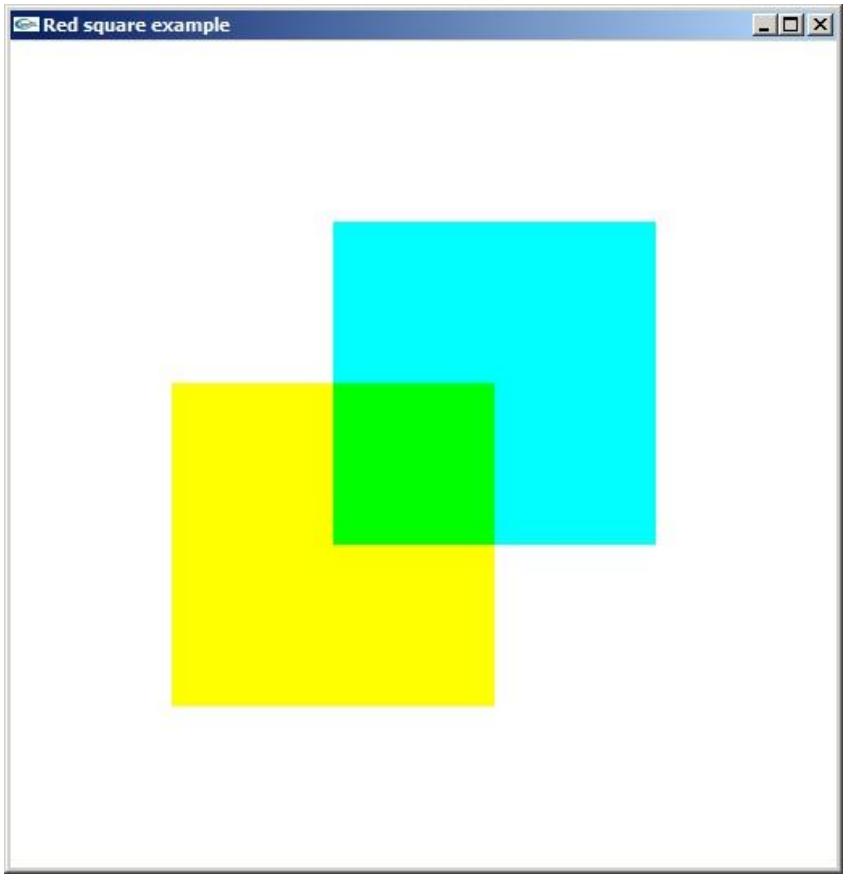
```
glRotatef ( 15, 0,0,1 )
```

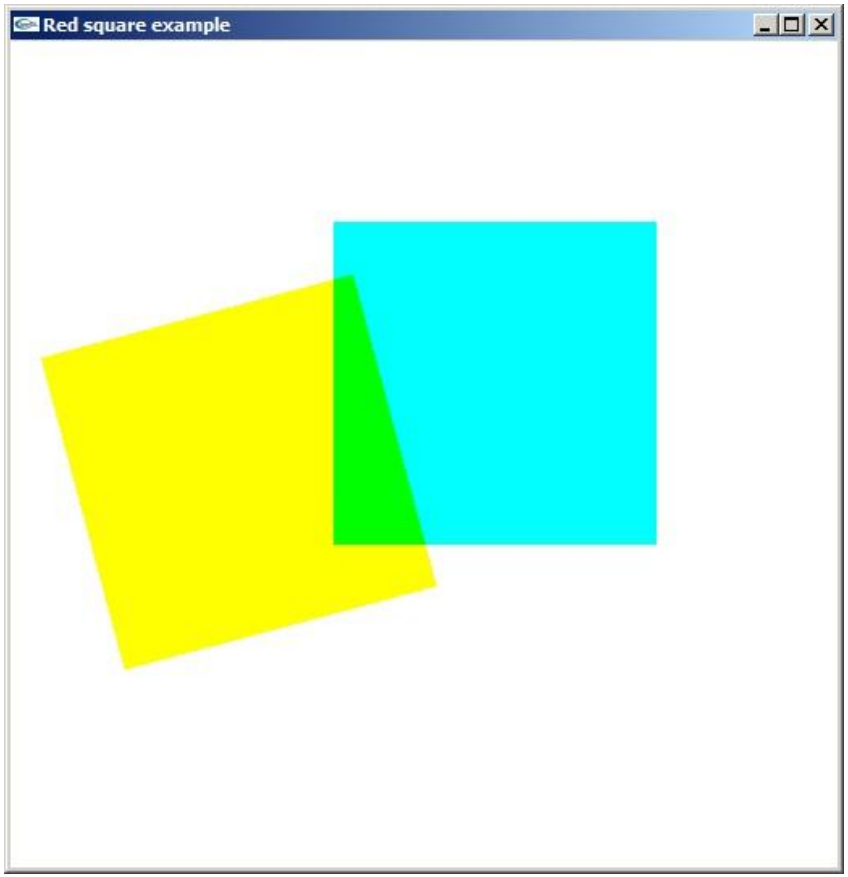




Модельно-видовые преобразования:
поворот

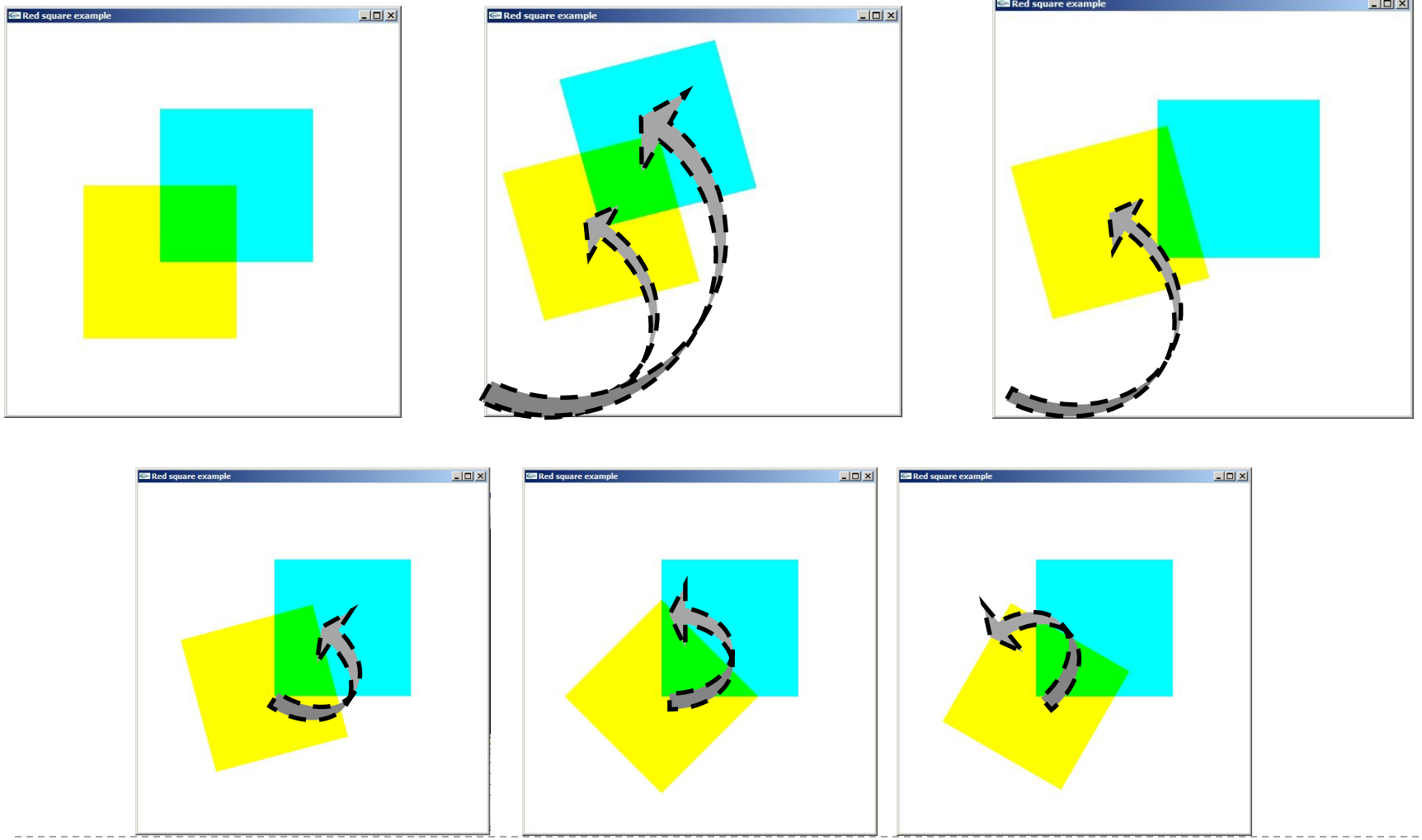
```
glRotatef ( 15, 0,0,1 )
```





Модельно-видовые преобразования:
поворот

```
glRotatef ( 15, 0,0,1 )
```



Модельно-видовые преобразования:
поворот

```
glRotatef ( 15, 0, 0, 1)

glColor3ub ( 255, 0, 0 );
glBegin ( GL_QUADS );
   glVertex2f ( 100, 100 );
   glVertex2f ( 100, 300 );
   glVertex2f ( 300, 300 );
   glVertex2f ( 300, 100 );



   glColor3ub ( 0, 255, 0 );

   glVertex2f ( 200, 200 );
   glVertex2f ( 200, 400 );
   glVertex2f ( 400, 400 );
   glVertex2f ( 400, 200 );
glEnd ();
```

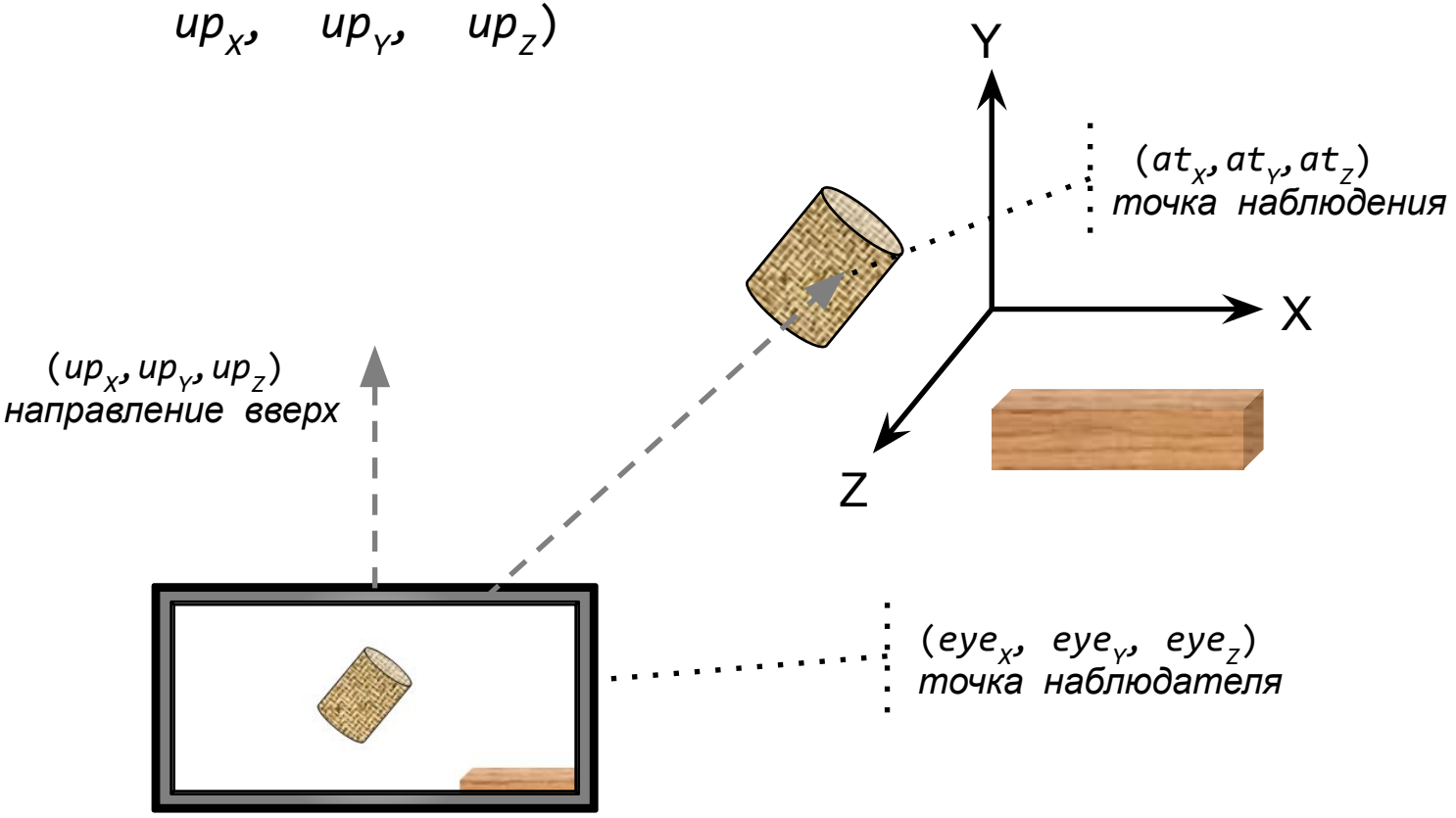
```
glRotatef ( 15, 0, 0, 1)

glColor3ub ( 255, 0, 0 );
glBegin ( GL_QUADS );
   glVertex2f ( 100, 100 );
   glVertex2f ( 100, 300 );
   glVertex2f ( 300, 300 );
   glVertex2f ( 300, 100 );
glEnd ();

glColor3ub ( 0, 255, 0 );
glBegin ( GL_QUADS );
   glVertex2f ( 200, 200 );
   glVertex2f ( 200, 400 );
   glVertex2f ( 400, 400 );
   glVertex2f ( 400, 200 );
glEnd ();
```

glPushMatrix();

glPopMatrix();

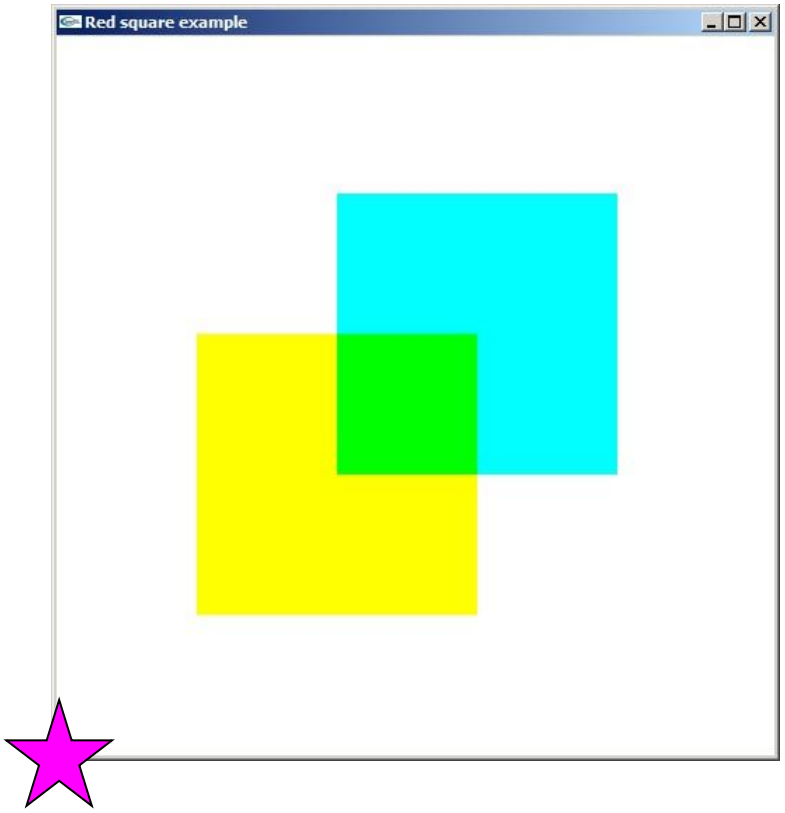
Модельно-видовые преобразования:
изменение положения камеры

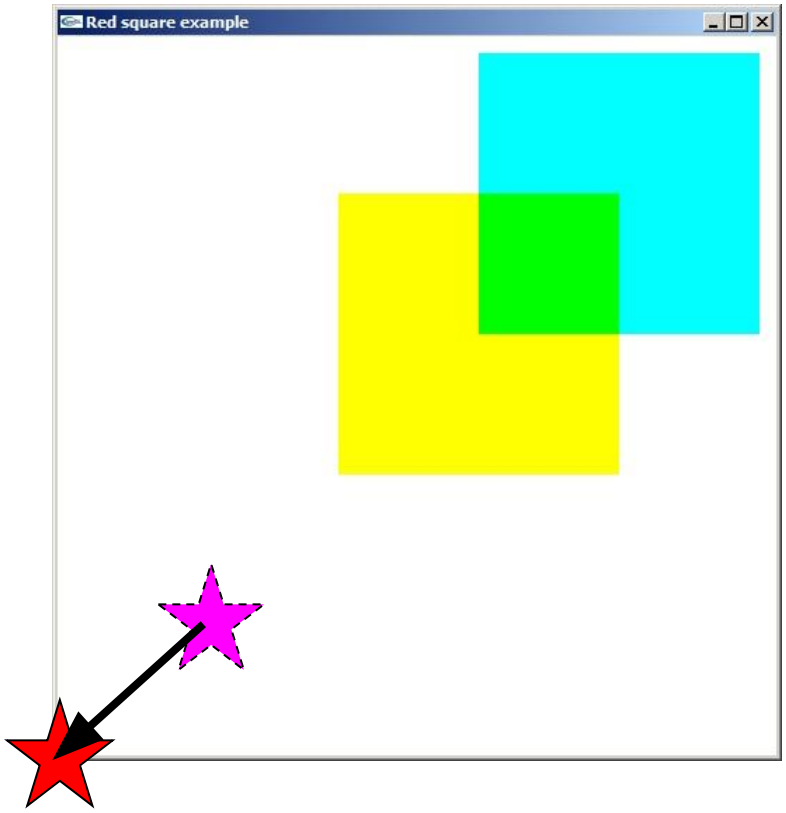
glu`LookAt` (eye_X, eye_Y, eye_Z,
at_X, at_Y, at_Z,
up_X, up_Y, up_Z)



Модельно-видовые преобразования:
изменение положения камеры

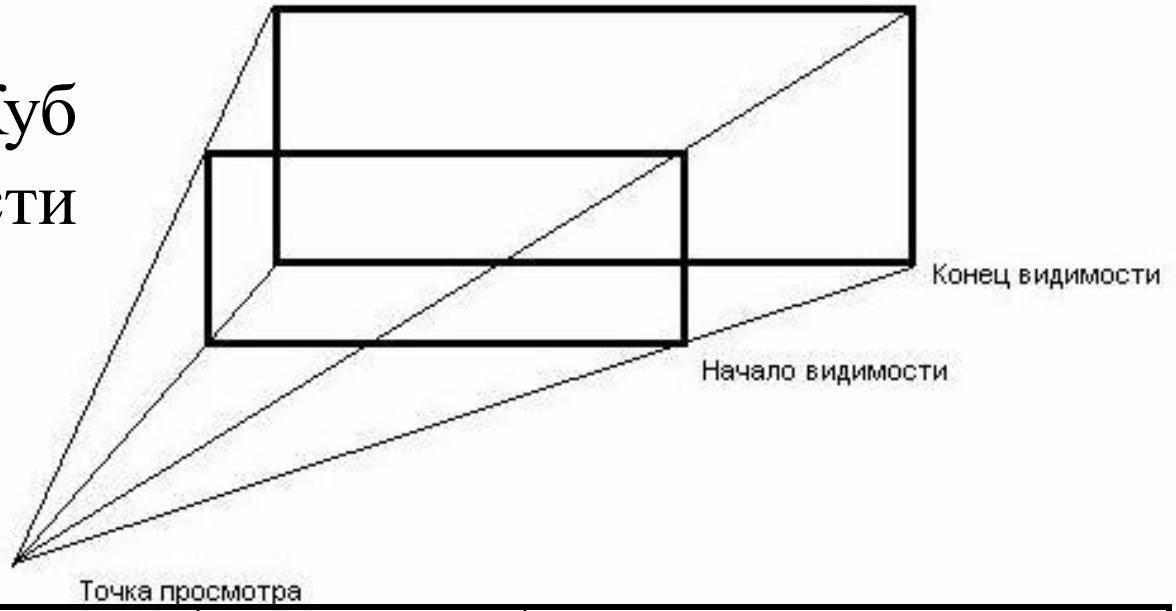
```
glTranslatef ( 100, 100, 10 )
gluLookAt (-100,-100,10, -100,-100,0, 0,1,0 )
```





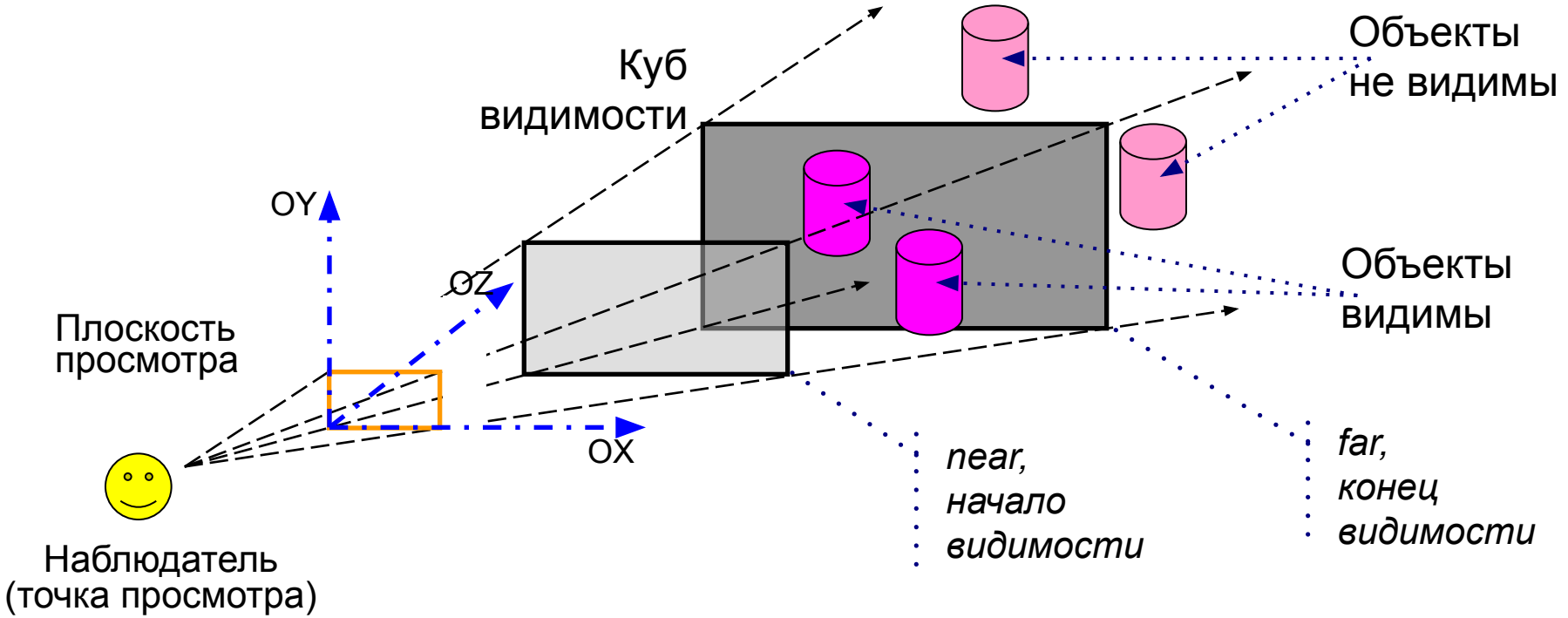
Преобразования проекции:
куб видимости

Куб видимости



Плоскости отсечения	*near*	Передняя (ближняя) по OZ
	far	Задняя (дальняя) по OZ
	left	Левая
	right	Правая
	top	Верхняя
	bottom	Нижняя

Преобразования проекции:
куб видимости



Плоскости отсечения (границы видимости, грани куба видимости)	*near*	Передняя (ближняя) по OZ
	far	Задняя (дальняя) по OZ
	left	Левая
	right	Правая
	top	Верхняя
	bottom	Нижняя

Преобразования проекции:
ортографическая (параллельная) проекция

- `glOrtho ( GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far )`

- `gluOrtho2D ( GLdouble left, GLdouble right, GLdouble bottom, GLdouble top )`

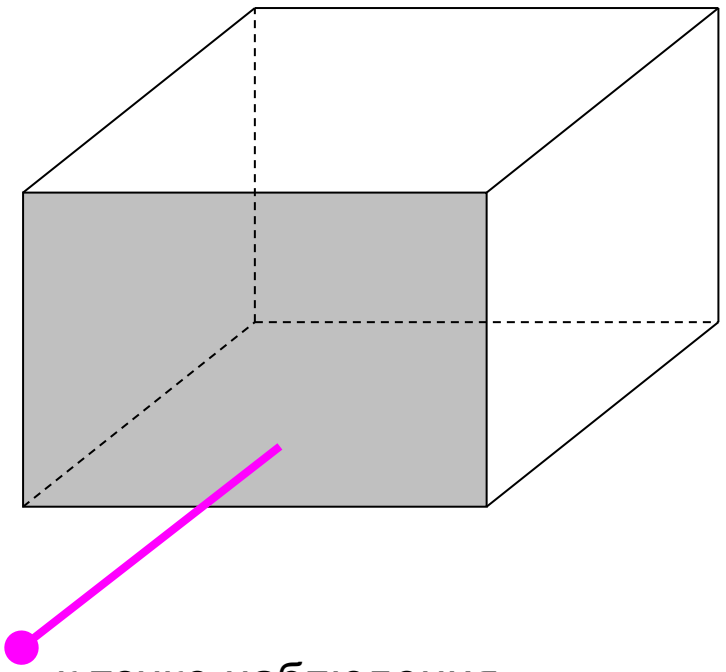
`gluOrtho2D ( left, right, bottom, top )`
≡
`glOrtho ( left, right, bottom, top, -1, 1 )`

Куб видимости	Параллелограмм
Система координат	Левосторонняя
(*left*, *bottom*, *-near*)	левый нижний угол окна вывода
(*right*, *top*, *-near*)	правый верхний угол окна вывода

Преобразования проекции:
ортографическая (параллельная) проекция

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glOrtho
    (
     left, right,
     bottom, top,
     near, far
    )
```



Преобразования проекции:
перспективная (центральная) проекция

- `gluPerspective ( GLdouble angle_y, GLdouble aspect, GLdouble near, GLdouble far )`

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

```
glFrustum ( -right, right,  -top, top, near, far)
≡
gluPerspective ( angle_y, aspect, near, far )
```

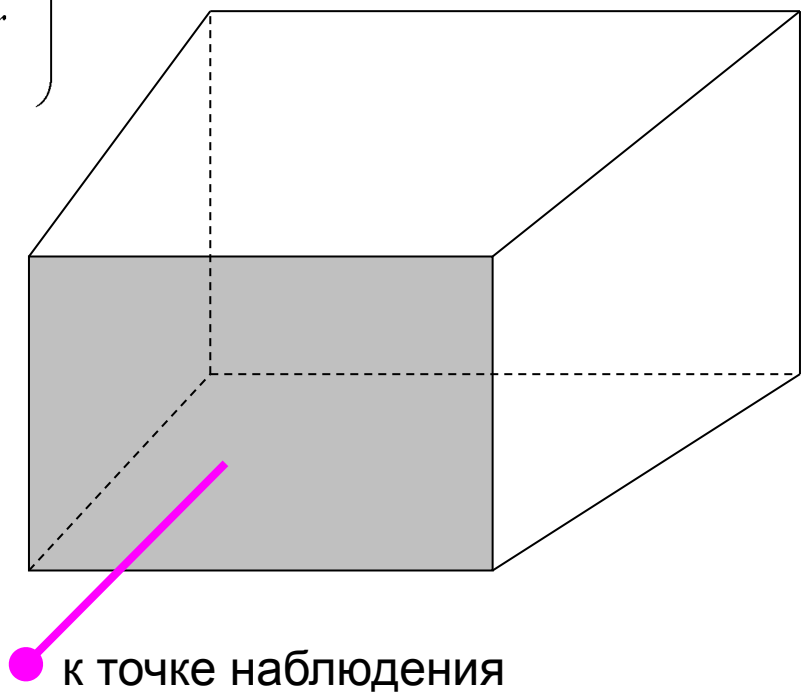
far > near > 0

Куб видимости	Усеченная пирамида
Система координат	Левосторонняя
angle_y	Угол видимости по OY (0° - 180°) = *2 * arctg(top / near)*
aspect	Угол видимости по OX = *right / top*

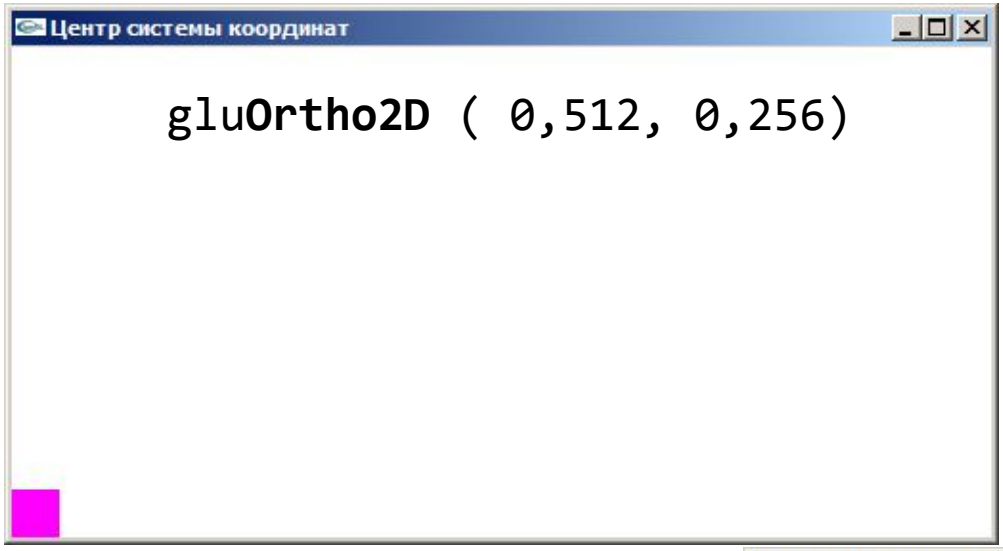
Преобразования проекции:
перспективная (центральная) проекция

$$\begin{pmatrix} \frac{2\times near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2\times near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2\times far\times near}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

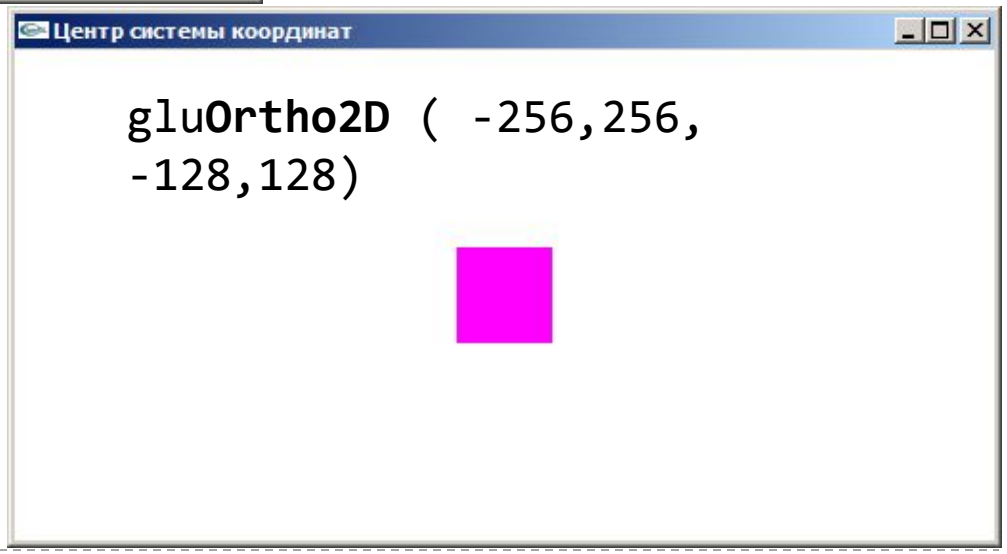
```
glFrustum
    (
     left, right,
     bottom, top,
     near, far
    )
```



Преобразования проекции:
задание центра системы координат



Центр системы координат задан в левом нижнем угле окна



Центр системы координат задан в центре окна

Дополнительные возможности отсечения

✔ Дополнительные плоскости отсечения в OpenGL (до 6 штук)

```
glClipPlane (*)

glEnable  ( GL_CLIP_PLANE plane )    // включить
glDisable ( GL_CLIP_PLANE plane )    // выключить
```

✔ Прямоугольник области вырезания
(команды могут модифицировать только пиксели,
лежащие внутри этой области)

```
glEnable  ( GL_SCISSOR_TEST ) // включить
glDisable ( GL_SCISSOR_TEST ) // выключить

glScissor ( x0,y0, width,height ) // область вырезания
```

Преобразование координат:
1. Мировые координаты

- На входе: координаты в локальных системах координат
  $(x_L, y_L, z_L)^T$

- Перевод в мировую (глобальную, правостороннюю) систему координат
  $(x_G, y_G, z_G)^T$

Преобразование координат:
2. Видовые координаты

- На входе: мировые координаты
  $(x, y, z)^T$
- Перевод в расширенные координаты:
  $(x, y, z, 1)^T$
- Модельно-видовые преобразования:
  $\mathbf{M} * (x, y, z, 1)^T = (x`, y`, z`, 1)^T$
- На выходе: видовые координаты
  $(x`, y`, z`, 1)^T$

Преобразование координат :
3. Нормализованные координаты

- На входе: видовые координаты
  $(x, y, z, w)^T$

- Преобразования проекции и отсечения:
  $\mathbf{M} * (x, y, z, w)^T = (x_c, y_c, z_c, w_c)^T$

- Преобразование нормализации:

$$\left(\frac{x_C}{w_C}, \frac{x_C}{w_C}, \frac{x_C}{w_C} \right)^T = (x_N, x_N, x_N)^T$$

На выходе: нормализованные координаты:

$$(x_N, x_N, x_N)^T$$

Преобразование координат :
4. Оконные координаты

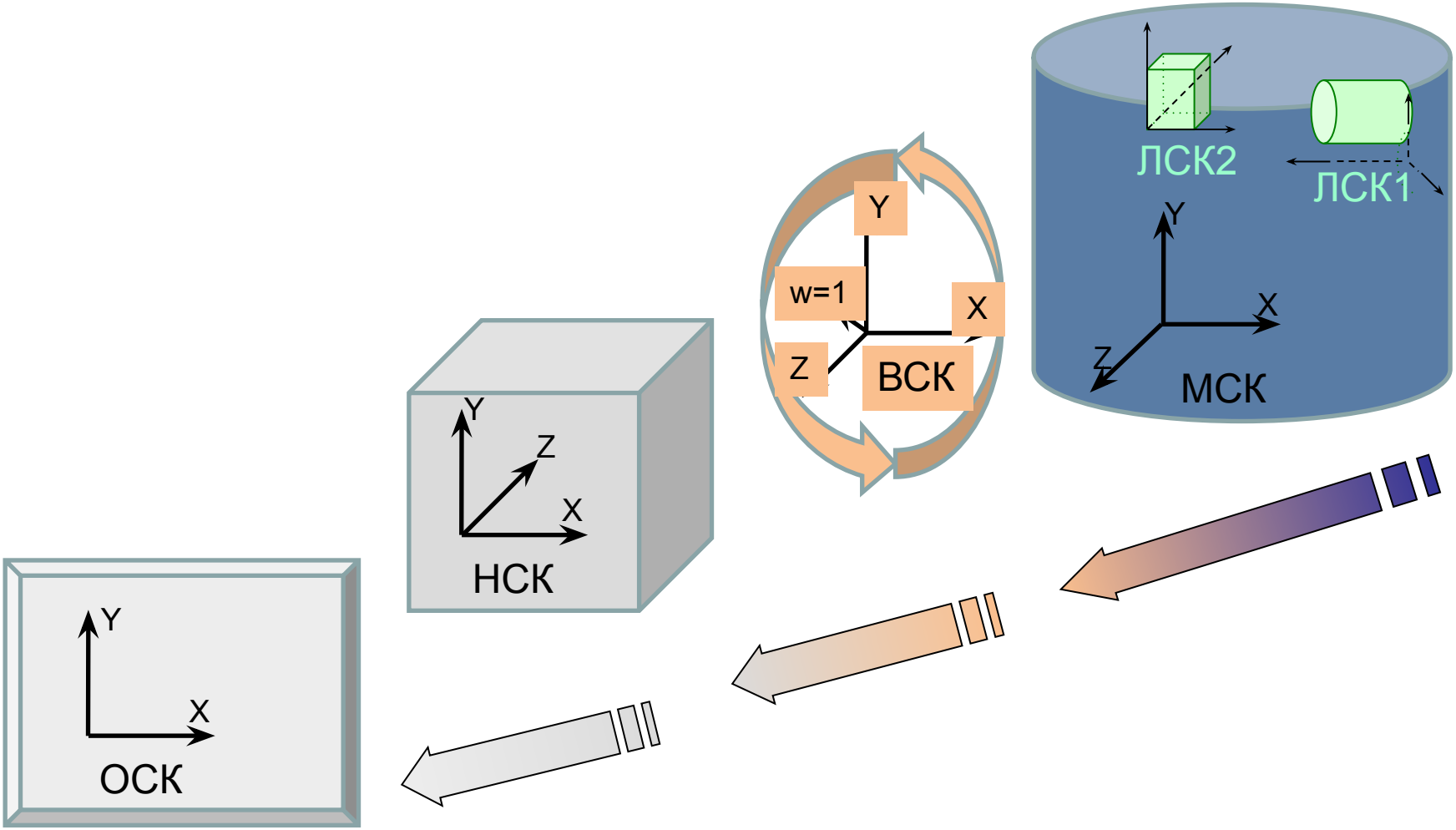
- На входе: нормализованные координаты $(x, y, z)^T$
- Определяется порт просмотра:
  `glViewPort(`x_P`, `y_P`, width, height)`
- Определяется параметры буфера глубины:
  `glDepthRange(near, far)`
- Расчет оконных координат $(x_w, y_w, z_w)^T$:

$$x_W = \left(\frac{\text{width}}{2} + 1\right) * x + x_P$$

$$y_W = \left(\frac{\text{height}}{2} + 1\right) * y + y_P$$

$$z_W = \left(\frac{far - near}{2}\right) * z + \left(\frac{far + near}{2}\right) \rightarrow 0$$

Преобразование координат :
конвейер



Матричные операции
в OpenGL

- Выбор матрицы: gl**MatrixMode**(*)
  - GL_MODELVIEW – модельно-видовая матрица
  - GL_PROJECTION – проекционная матрица
  - GL_TEXTURE – текстурная матрица
- Сохранение матрицы в массив 4x4 (16 элементов) по столбцам:
  - gl**GetFloat**v (GL_MODELVIEW_MATRIX, *matr*)
- Загрузка матрицы из массива: gl**LoadMatrix**(*matr*)
- Сброс матрицы: gl**LoadIdentity**()
- Умножение матрицы слева: gl**MultMatrix**(*matr*)
- Сохранение матрицы в стеке: gl**PushMatrix**()
- Восстановление матрицы из стека: gl**PopMatrix**()