

**Смешивание**  
**Сглаживание**  
**Туман**  
**Параметры точки**

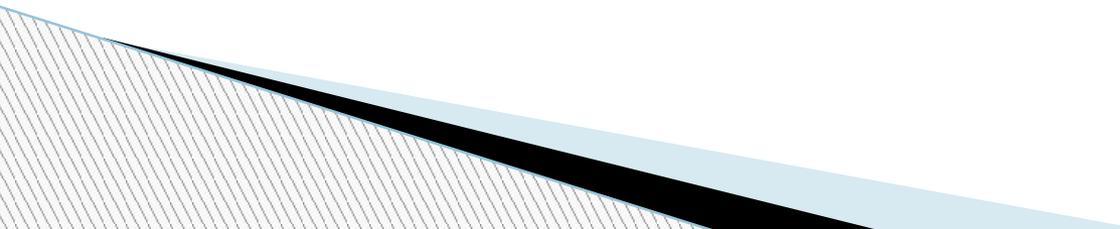


# Смешивание

- ▣ Что такое альфа
- ▶ Факторы влияния источника (source) и получателя (destination)
  - Указываем факторы
  - Комбинирование значений
$$(C_s S_c + C_d D_c), C = R, G, B, A$$
  - Ограничивание диапазоном [0, 1]

# Коэффициенты смешивания

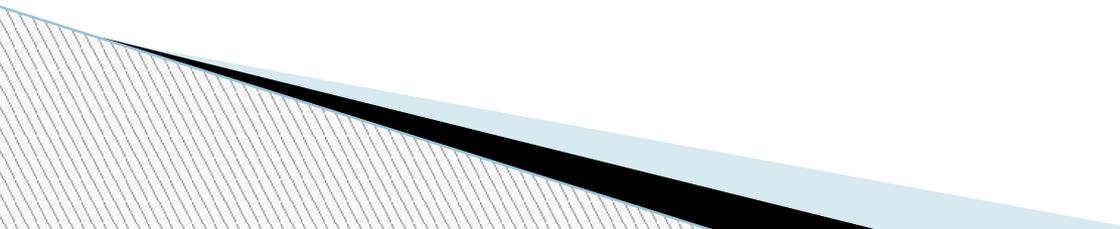
```
void glBlendFunc(  
GLenum srcfactor,  
GLenum destfactor  
);
```



Константа	Коэффициент для RGB	Коэффициент для А
GL_ZERO	0	0
GL_ONE	1	1
GL_SRC_COLOR		
GL_ONE_MINUS_SRC_COLOR		
GL_DST_COLOR		
GL_ONE_MINUS_DST_COLOR		
GL_SRC_ALPHA		
GL_ONE_MINUS_SRC_ALPHA		
GL_DST_ALPHA		
GL_ONE_MINUS_DST_ALPHA		
GL_CONSTANT_COLOR		
GL_ONE_MINUS_CONSTANT_COLOR		
GL_CONSTANT_ALPHA		
GL_ONE_MINUS_CONSTANT_ALPHA		
GL_SRC_ALPHA_SATURATE		1

# GL\*CONSTANT\*

```
void glBlendColor(  
GLclampf red,  
GLclampf green,  
GLclampf blue,  
GLclampf alpha  
);
```



# Включения смешивания

```
glEnable (GL_BLEND);  
glDisable (GL_BLEND); /*  
    * S: GL_ONE,  
    * D: GL_ZERO  
    */
```

# Объединение пикселей на основе выражений

`void glBlendEquation (GLenum mode);`

Параметр	Операция
GL_FUNC_ADD	
GL_FUNC_SUBTRACT	
GL_FUNC_REVERSE_SUBTRACT	
GL_MIN	
GL_MAX	

# Трёхмерное смешивание

Буффер глубины в режиме «только для чтения»

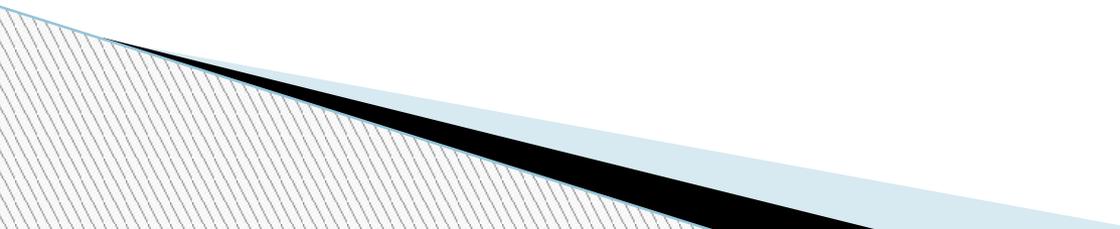
`glDepthMask();`

`GL_FALSE` – только для чтения

`GL_TRUE` – нормальный режим

# Сглаживание

```
void glHint (  
GLenum target,  
GLenum hint    /*  
    * GL_FASTER  
    * GL_NICEST  
    * GL_DONT_CARE  
    */  
);
```



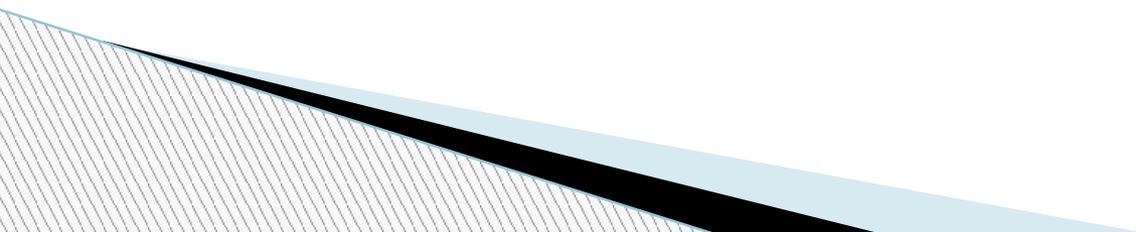
Значение target	Применение
GL_POINT_SMOOTH_HINT GL_LINE_SMOOTH_HINT GL_POLYGON_SMOOTH_HINT	Управление качеством сглаживания точек, линий, полигонов
GL_FOG_HINT	Указывает, выполнять ли вычисления при наложении тумана по пикселям или по вершинам
GL_PERSPECTIVE_CORRECTION_HINT	Качество интерполяции координат при вычислении цветов и наложения текстур
GL_GENERATE_MIPMAP_HINT	Качество и производительность автоматической генерации MIPMAP-текстур
GL_TEXTURE_COMPRESSION_HINT	Качество и производительность сжатых текстурных изображений

# Пример сглаживания в режиме RGBA

- ▣ glEnable ()
  - GL\_POINT\_SMOOTH
  - GL\_LINE\_SMOOTH
  - GL\_POLYGON\_SMOOTH
- ▣ glEnable (GL\_BLEND)
- ▣ glBlendFunc (GL\_SRC\_ALPHA,)
  - GL\_ONE\_MINUS\_SRC\_ALPHA
  - GL\_ONE

# Туман

`glEnable (GL_FOG);`



# Уравнения тумана

- $f = e^{-(density*z)}, (GL\_EXP)$
- $f = e^{-(density*z)^2}, (GL\_EXP2)$
- $f = \frac{end - z}{end - start}, (GL\_LINEAR)$

# Туман в режиме RGBA

```
void glFog{i,f}[v](GLenum pname,  
                    /*GL_FOG_MODE  
                    * GL_FOG_DENSITY  
                    * GL_FOG_START  
                    * GL_FOG_END  
                    * GL_FOG_COLOR  
                    */  
                    TYPE [*]param);
```

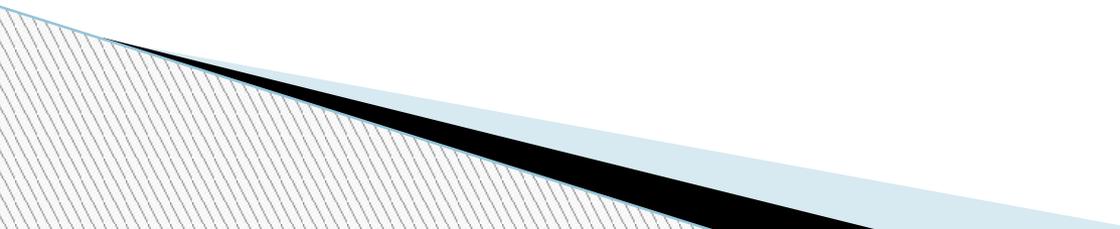
$$C = fC_i + (1 - f)C_{fi}$$

# Координаты тумана

```
glFog (GL_FOG_COORDINATE_SOURCE,  
       GL_FOG_COORDINATE);  
void glFogCoord{f,d}[v] (TYPE [*]z);  
// z > 0
```

# Параметры точки

```
void glVertexParameterf(v)(
GLenum pname,
/* GL_POINT_DISTANCE_ATTENUATION,
 * GL_POINT_SIZE_MIN,
 * GL_POINT_SIZE_MAX */
GLfloat [*]param);
```



# Уравнение точки

$$\square \quad \textit{derivedSize} = \textit{clamp} \left( \textit{size} * \sqrt{\frac{1}{a + b * d + c * d^2}} \right)$$

```
glEnable (GL_POINT_SMOOTH);
```

```
glEnable (GL_BLEND);
```

```
glBlendFunc (GL_SRC_ALPHA,  
             GL_ONE_MINUS_SRC_ALPHA);
```