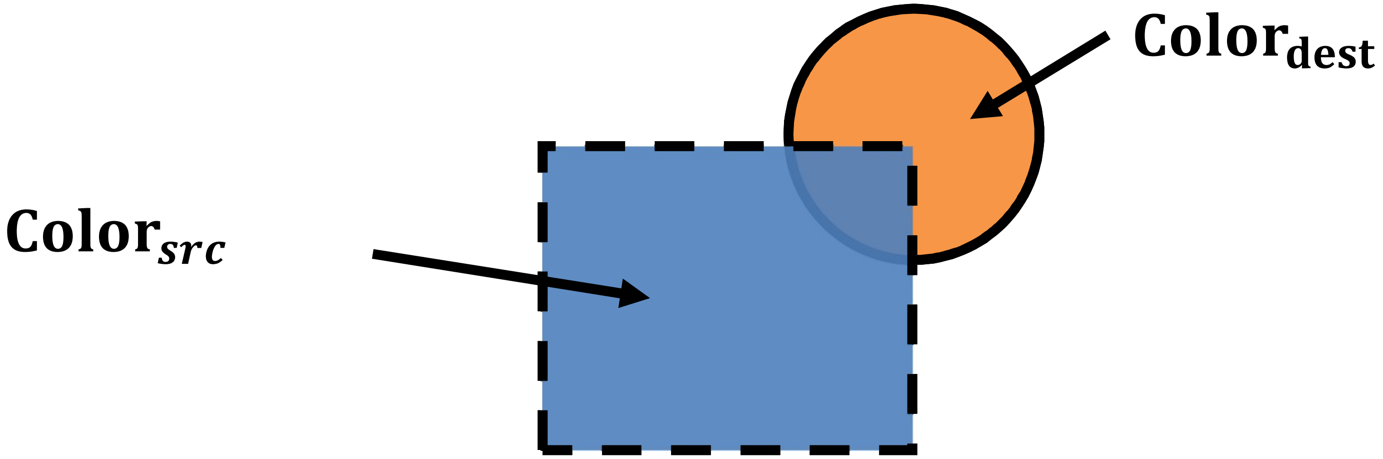
OpenGL – Alpha and Blending

Transparency – Formula (OpenGL Naming)

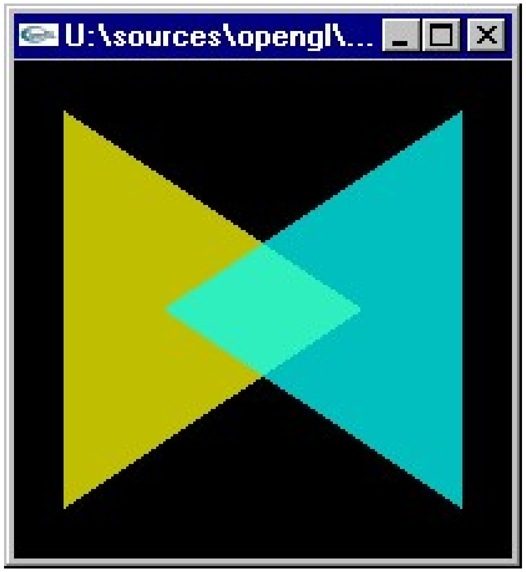
- $\mathbf{Color}_{\boldsymbol{out}} = \mathbf{Color}_{\boldsymbol{dest}} * (1 - alpha_{src}) + \mathbf{Color}_{\boldsymbol{src}} * alpha_{src}$



$\mathbf{Color}_{\mathbf{dest}}$

$\mathbf{Color}_{\boldsymbol{src}}$

Alpha



- RGB(Alpha) = RGBA, A=α (opacity)
  - **`glColor4f(1, 0, 0, 1);`**
  - $\alpha = 1$ means opaque
  - **`glColor4f(1, 0, 0, 0);`**
  - $\alpha = 0$ means completely transparent (like air)
  - **`glColor4f(1, 0, 0, 0.5);`**
  - $\alpha = 0.5$ means semi transparent
- But alpha does nothing without blending

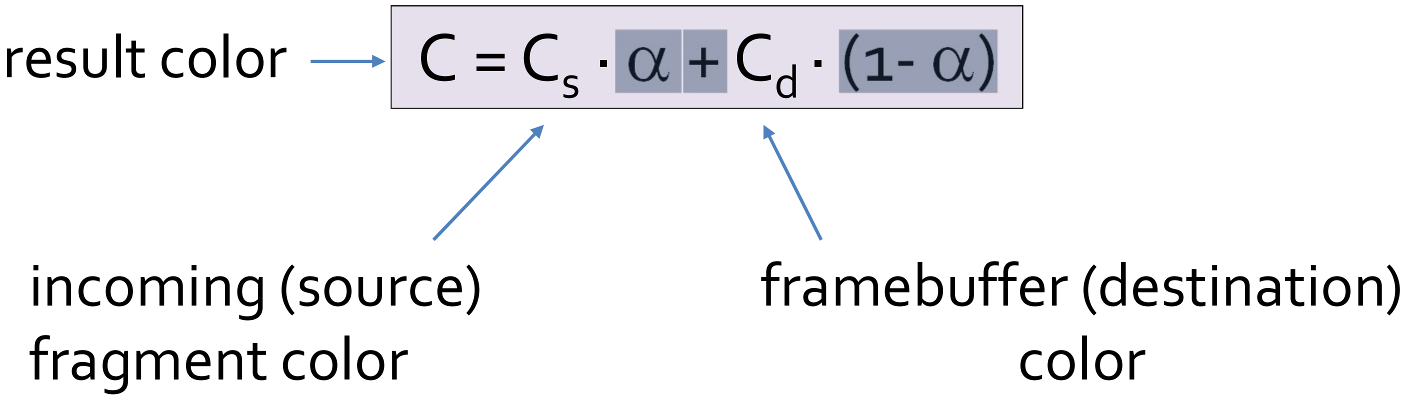
Blending

- **`glEnable(GL_BLEND);`**
- Outside `glBegin/glEnd` pair!
- **`glBlendFunc(<SRC>,<DST>); glBlendEquation(<OP>);`**
- No blending: color => framebuffer
- With blending: combination of source and destination color

Blending

- Example: transparency blending (window)

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glBlendEquation(GL_FUNC_ADD);
```



result color

$$C = C_s \cdot \alpha + C_d \cdot (1 - \alpha)$$

incoming (source) fragment color

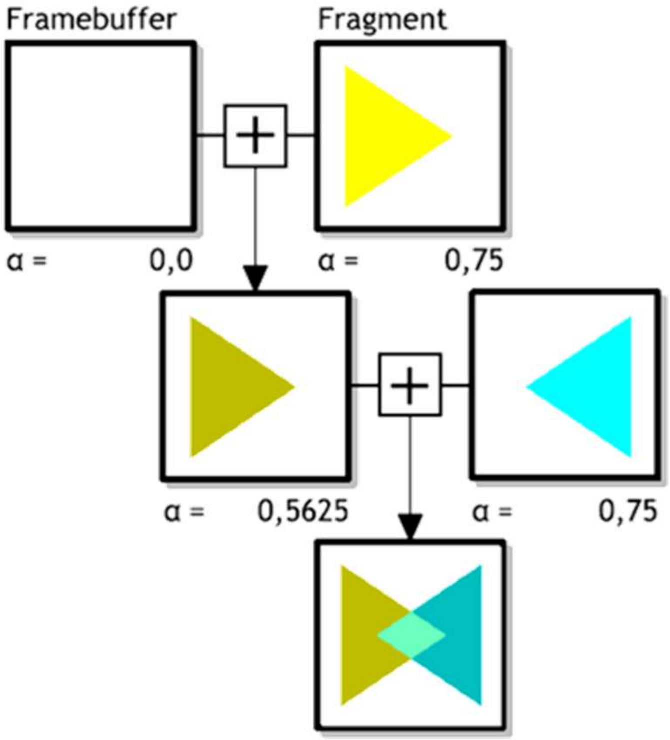
framebuffer (destination) color

Blending Functions

Enum	Factor	Calcul. Factor
GL_ZERO	s/d	(0,0,0,0)
GL_ONE	s/d	(1,1,1,1)
GL_DST_COLOR	source	(Rd,Gd,Bd,Ad)
GL_SRC_COLOR	destination	(Rs,Gs,Bs,As)
GL_ONE_MINUS_DST_COLOR	source	(1,1,1,1)-(Rd,Gd,Bd,Ad)
GL_ONE_MINUS_SRC_COLOR	destination	(1,1,1,1)-(Rs,Gs,Bs,As)
GL_SRC_ALPHA	s/d	(As,As,As,As)
GL_ONE_MINUS_SRC_ALPHA	s/d	(1,1,1,1)-(As,As,As,As)
GL_DST_ALPHA	s/d	(Ad,Ad,Ad,Ad)
GL_ONE_MINUS_DST_ALPHA	s/d	(1,1,1,1)-(Ad,Ad,Ad,Ad)

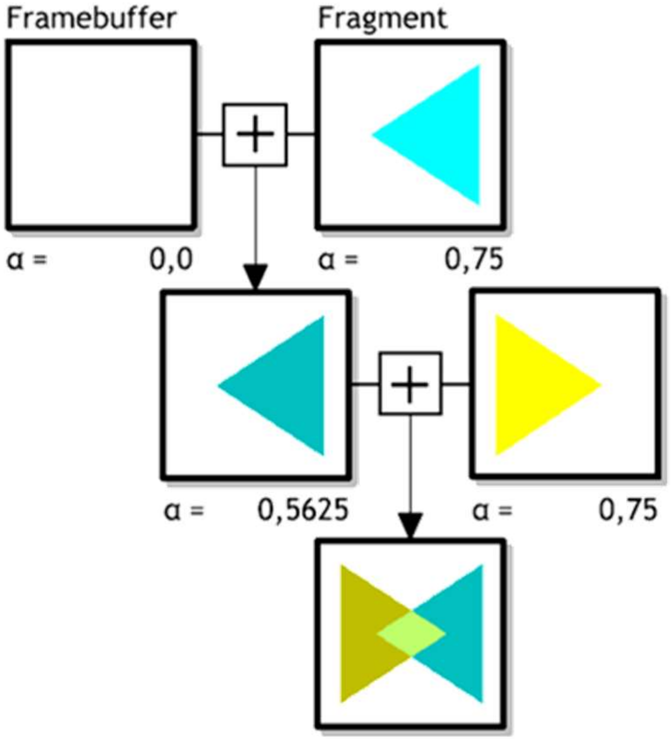
Example 1

```
//Initialize alpha blending function.
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_BLEND);

glColor4f(1, 1, 0, 0.75);
drawLeftTriangle();
glColor4f(0, 1, 1, 0.75);
drawRightTriangle();
```

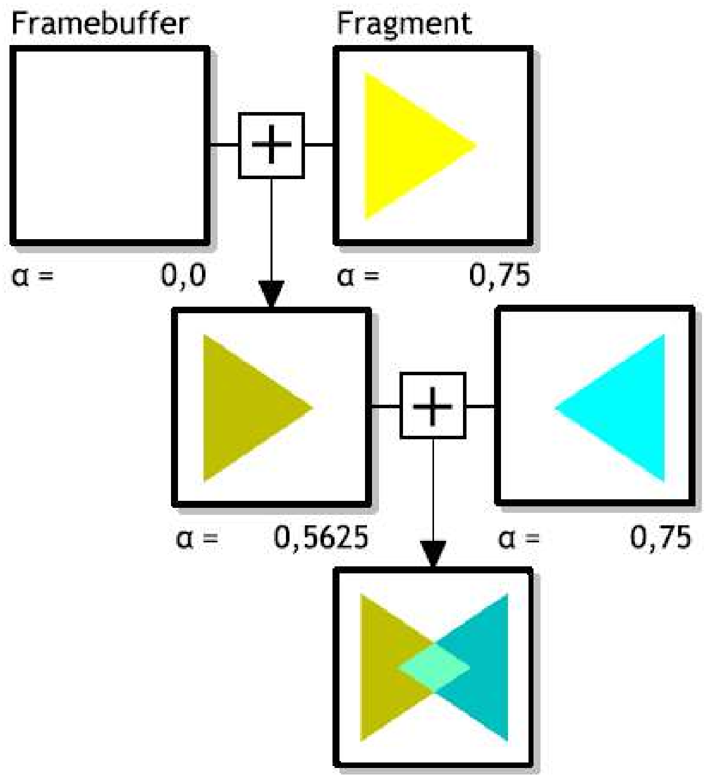


Example 1 Reverse drawing

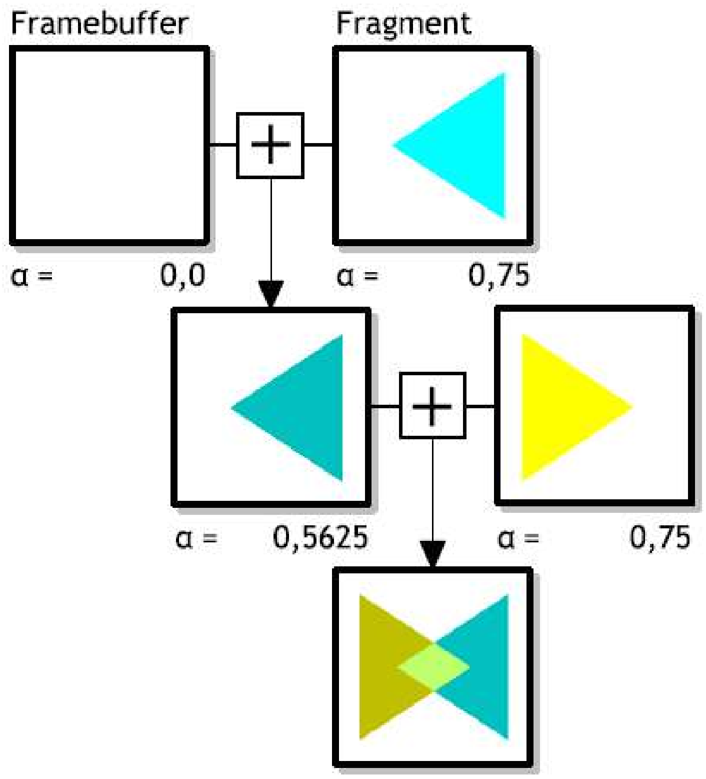
```
//Initialize alpha blending function.
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_BLEND);

glColor4f(0, 1, 1, 0.75);
drawRightTriangle();
glColor4f(1, 1, 0, 0.75);
drawLeftTriangle();
```



Both Examples



(a) Gelb → Cyan



(b) Cyan → Gelb

Alpha Test

- Accept/reject fragments based on alpha

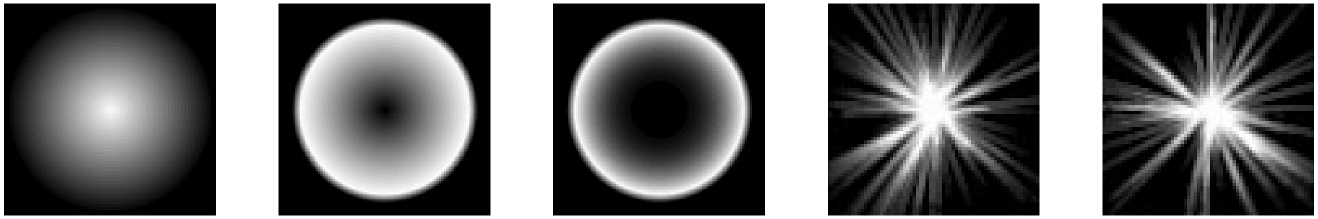
```
glEnable (GL_ALPHA_TEST)
glDisable (GL_ALPHA_TEST)
glAlphaFunc(GLenum func, GLclampf ref)
```

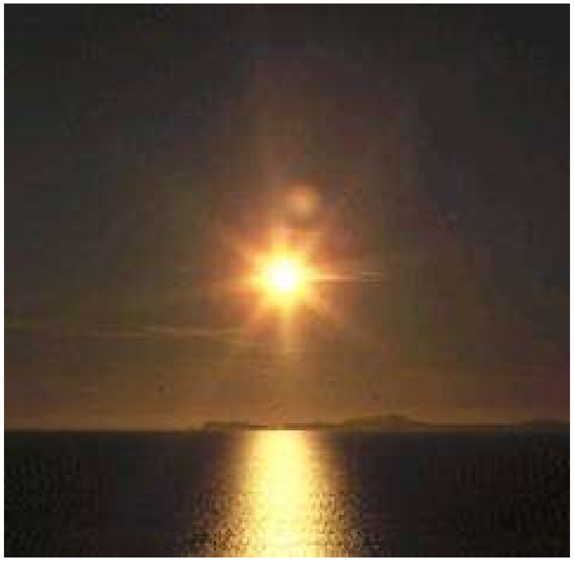
Alpha Test Function

Function	Meaning
`GL_NEVER`	`never accept the fragment`
`GL_ALWAYS`	`always accept the fragment`
`GL_LESS`	`accept fragment if its alpha < reference alpha`
`GL_LEQUAL`	`accept fragment if its alpha <= reference alpha`
`GL_EQUAL`	`accept fragment if its alpha = reference alpha`
`GL_GEQUAL`	`accept fragment if its alpha >= reference alpha`
`GL_GREATER`	`accept fragment if its alpha > reference alpha`
`GL_NOTEQUAL`	`accept fragment if its alpha != reference alpha`

```
e.g., glAlphaFunc(GL_GREATER, 0.5);
```

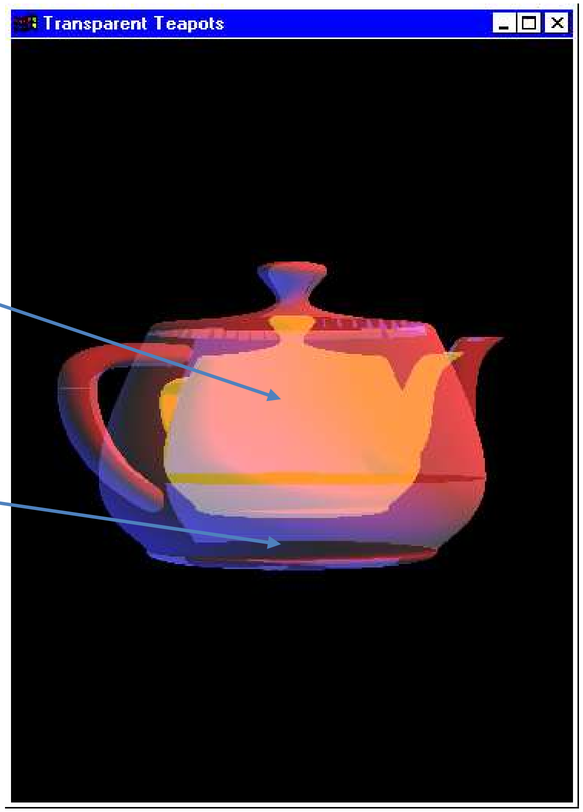
Lens Flare Example





3D Blending Example

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

glDisable(GL_BLEND);
glColor3f(1.0, 0.0, 0.0);
glutSolidTeapot(0.4);

glEnable(GL_BLEND);
glColor4f(0.4, 0.0, 1.0, 0.25);
glutSolidTeapot(0.6);
```



Blending and Z-Buffer

- Be careful: What do we know about the z-buffer test?
- What do we know about blending?

→ So what?

Blending and Z-Buffer - Solution

- Transparent objects rendered in **front** of opaque objects
- Z-buffer read-only for transparent objects
  - **`glDepthMask(boolean)`**
- Sort blended objects