3D Graphics Programming

T163 - Game Programming

Week 13

Blending Basics

Blending

- The first way I showed transparency is not ideal because the texture doesn't end up looking that great
- In the fragment shader, we can put this:

```
1   #version 430 core
2
3   in vec3 colour;
4   in vec2 texCoord;
5   out vec4 frag_colour;
6
7   uniform sampler2D texture0;
8
9   void main()
10  {
11      vec4 texColor = texture(texture0, texCoord) * vec4(colour, 1.0f);
12      if(texColor.a < 0.1)
13          discard;
14      frag_colour = texColor;
15  }
```

- Note that the frag_colour MUST go out as a vec4
- All the examples should have this already

Blending

- The second way is to enable blending in OpenGL's rendering pipeline:

```
// Enable blending.
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

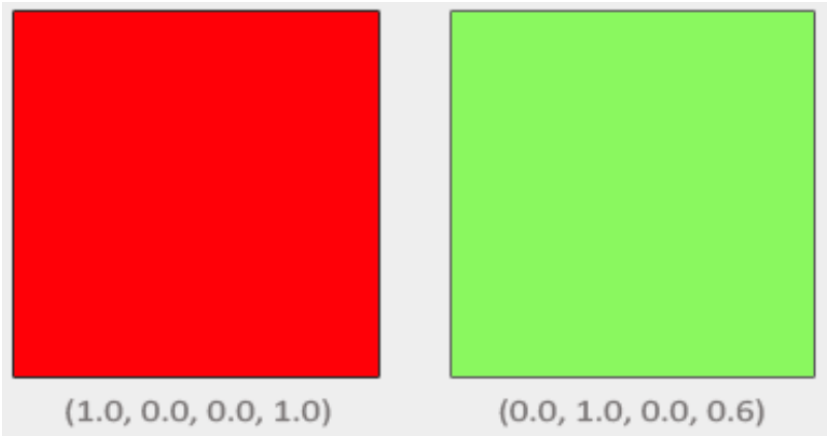
- This makes transparencies look much better than the 'discard' method
- But what's that glBlendFunc and its arguments? Well, it's this:

$$(R_{d'}, G_{d'}, B_{d'}, \alpha_{d'}) = (\alpha_s R_s + (1-\alpha_s)R_d, \alpha_s G + (1-\alpha_s)G_d, \alpha_s B_s + (1-\alpha_s)B_d, \alpha_s \alpha_d + (1-\alpha_s)\alpha_d).$$

- GL_SRC_ALPHA is the source blending factor and GL_ONE_MINUS_SRC_ALPHA is the destination factor and the resulting formula ensures colors and opacities cannot saturate

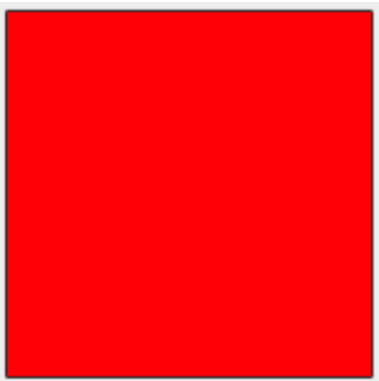
Blending

- Let's see an example!
- Simplifying our equation to: $\bar{C}_{result} = \bar{C}_{source} * F_{source} + \bar{C}_{destination} * F_{destination}$

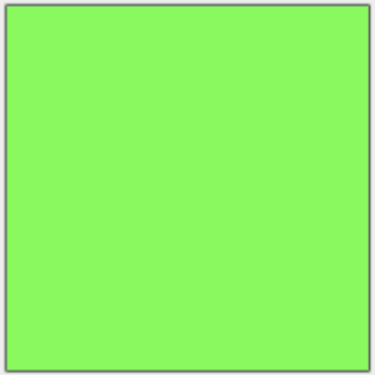


Blending

- Let's see an example!
- Simplifying our equation to: $\bar{C}_{result} = \bar{C}_{source} * F_{source} + \bar{C}_{destination} * F_{destination}$
- Destination



(1.0, 0.0, 0.0, 1.0)



(0.0, 1.0, 0.0, 0.6)

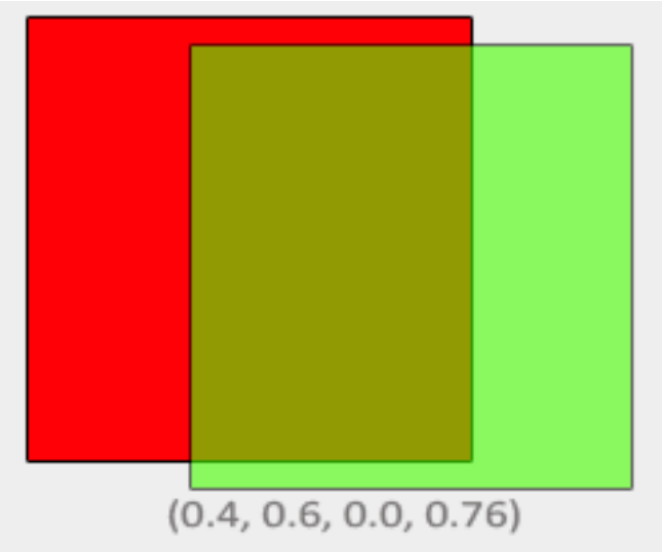
Source

- The equation becomes:

$$\bar{C}_{result} = \begin{pmatrix} 0.0 \\ 1.0 \\ 0.0 \\ 0.6 \end{pmatrix} * 0.6 + \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{pmatrix} * (1 - 0.6)$$

Blending

- And we get a blending like this:



Blending

- Some final notes on blending
- Draw the farthest object first and the closest object as last
- When mixing opaque and transparent objects:
  - Draw all opaque objects first
  - Sort all the transparent objects
  - Draw all the transparent objects in sorted order

  - You can sort the objects by distance from the camera

Blending

- Antialiasing
  - A smoothing effect of a line
  - Like most things in OpenGL, we have function calls:

```
glEnable(GL_LINE_SMOOTH);
glEnable(GL_POLYGON_SMOOTH);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Week 13

Lab Activities

Week 13 Lab

- For the lab, see Hooman's material (with video)

Week 13

End