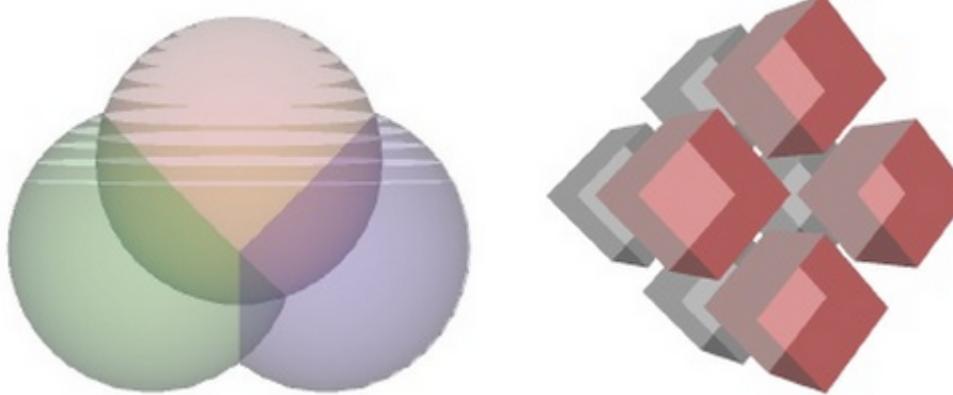

OpenGLで Order Independent Transparency

ペりむ (@hi2p_perim)

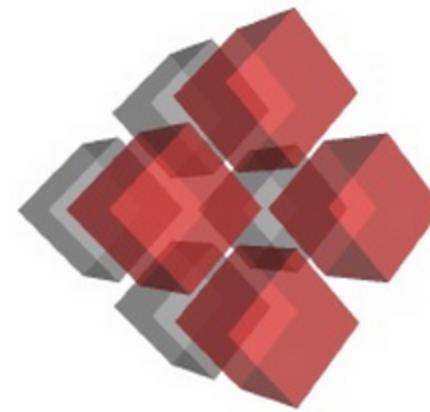
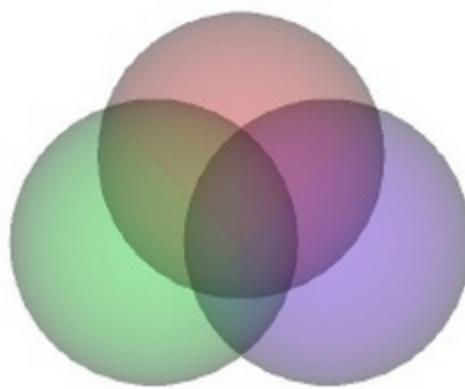
導入

OpenGLで半透明なオブジェクトを描画したい…
単純に glEnable(GL_BLEND) すると？



何かがおかしい！？

導入



正しくは…

なぜ？

アルファブレンディングの式

$$\mathbf{c}_o = \alpha_s \mathbf{c}_s + (1 - \alpha_s) \mathbf{c}_d$$

ブレンディング 後の色 描画するオブジェクトの色 描画するオブジェクトのアルファ ブレンディング 前の色

で得られる色は描画順序に依存する！

やること

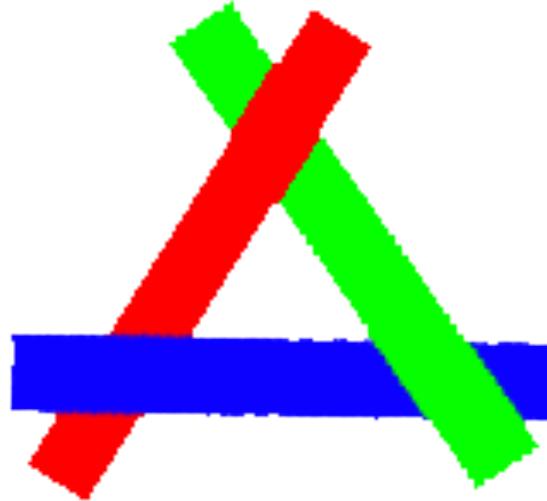
- Depth Sorting
- Depth Peeling
- Weighted Average
- OIT with Layered Fragment Buffer

Depth Sorting [Newell 1972]

基本：プリミティブ（三角形とか）をカメラからの距離が遠い順に描画する

Depth Sorting の問題点

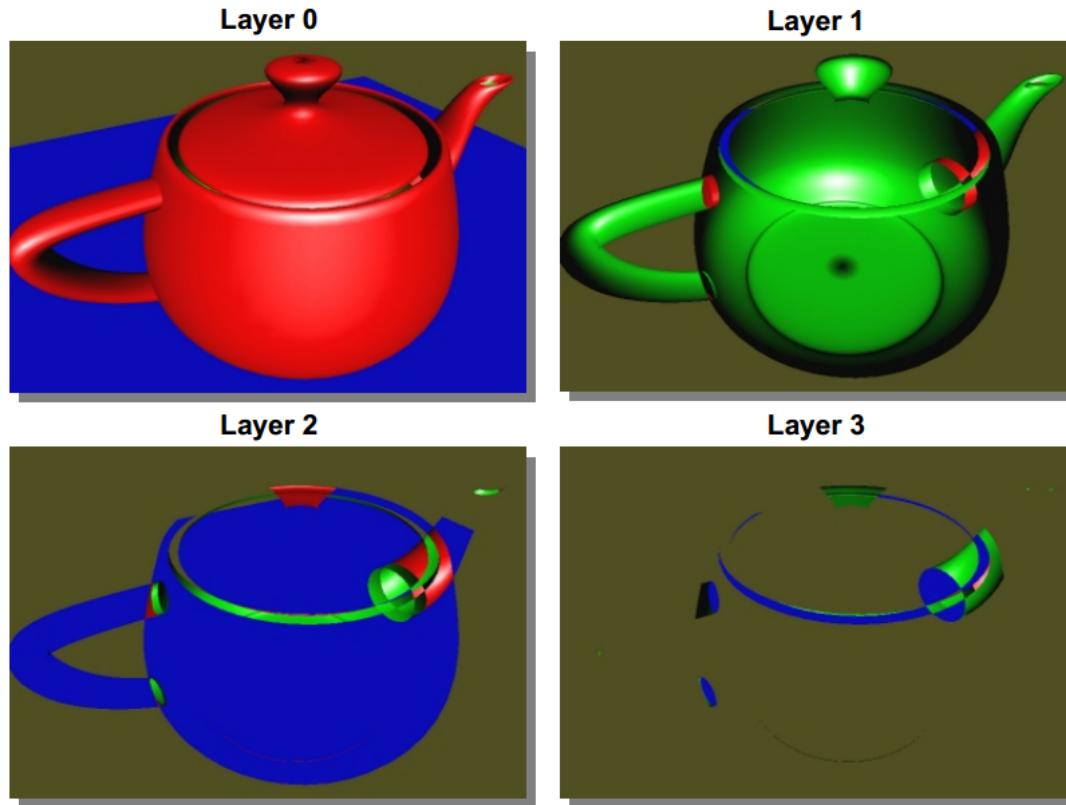
- ふつうにソートをCPUで行うと遅い
- プリミティブレベルでソートを行うので、あいまいなケースの対応がめんどい



あいまいなケースの例

Depth Peeling [Everitt 2001]

基本 : depthが異なる層の色を得る。



Depth Peeling

- depthを保持するテクスチャを-1で初期化
 - GL_R32Fでいい
- 次のようなfragment shader

```
uniform sampler2D depthMap;
uniform sampler2D prevColorMap;
layout (location = 0, index = 0) out vec4 fragColor;
layout (location = 1, index = 0) out vec4 fragDepth;
void main()
{
    float depth = texture(depthMap, gl_FragCoord.st).r;
    if (gl_fragCoord.z <= depth) discard;
    fragDepth.r = depth;
    vec4 prevColor = texture(prevColorMap, gl_FragCoord.st);
    fragColor.rgb = prevColor.rgb + (source color) * (source alpha) + prevColor.a;
    fragColor.a = (1.0 - (source alpha)) * prevColor.a;
}
```

Depth Peelingの問題点

- 複雑なシーンだと遅い
 - パスの回数はそれぞれのピクセルにおけるプリミティブの重なりの回数の最大)

Depth Peeling

改良手法いろいろ:

- Dual Depth Peeling [Bavoil 2008]
- Depth Peeling via Bucket Sort [Liu 2009]

Weighted Average [Meshkin 2007]

- 近似手法だが、結構うまくいく

OIT with Layered Fragment Buffer [Knowles 2012]

- 基本:ピクセルごとに重なっているプリミティブの情報にアクセスできればよいのでは?
- cf. A-buffer [Carpenter 1984]
- cf. DirectX11 version [Gruen 2010]



OIT with Layered Fragment Buffer

- Linked-list LFB
- Linear LFB
- OpenGL4が必要
 - GL_ARB_shader_image_load_store
 - シェーダ中からテクスチャに読み書きできる
 - GL_ARB_shader_atomic_counter
 - atomic counter
 - これがないとハザードが起こる

参考文献

- Everitt, C. 2001. Interactive Order Independent Transparency. NVIDIA Corporation.
- Bavoil, L. 2008. Order Independent Transparency with Dual Depth Peeling. NVIDIA Corporation.
- Meshkin, H. 2007. Sort-Independent Alpha Blending. Perpetual Entertainment, GDC 2007 Session.
- Newell, M. E. 1972. A Solution to the Hidden Surface Problem. In *Proceedings of the ACM National Conference 1972*, 443-450.
- Carpenter, L. 1984. The A-buffer, an antialiased hidden surface method. In *Proceedings of SIGGRAPH '84*.

参考文献

- Knowles, P., Leach, G., and Zembetta, F. 2012. Efficient Layered Fragment Buffer Techniques. In *OpenGL Insights*, CRC Press, 279-292.
- Gruen, H., and Thibieroz, N. 2010. OIT and Indirect Illumination using DX11 Linked Lists. AMD ISV Relations, GDC 2010 Session.
- Liu, F., Huang, M., Liu, X., and Wu. E. 2009. Single Pass Depth Peeling via Bucket Sort. In *Proceedings of HPG '09*, 51-57.