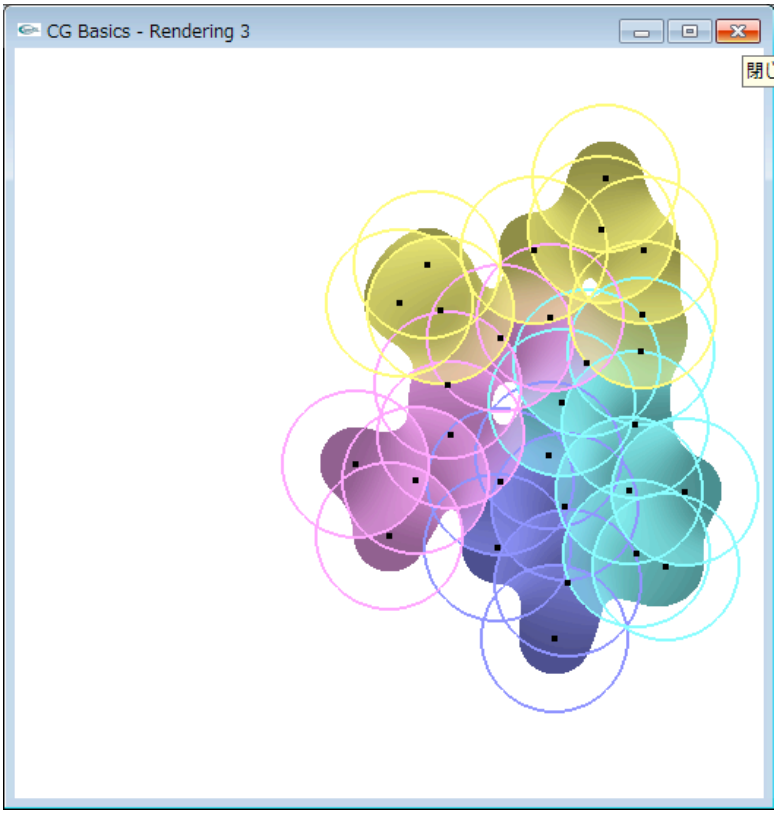
```
inline float CalcMetaballDensity(const Metaball2D *m, const float p[2])
{
    float r = Vec2Distance(m->m_Center, p);
    float R = m->m_Radius;
    float result = 0;
    if (r <= R) {
        result = -(4.0/9.0)*powf(r/R, 6) + (17.0/9.0)*powf(r/R, 4) - (22.0/9.0)*powf(r/R, 2) + 1;
    }
    return result;
}

void UpdateMetaballBuffer()
{
    /* 省略 */
    for (int i=0; i<g_NumMetaballs; i++) {
        const Metaball2D *m = &g_Metaballs[i];
        const float intensity = m->m_Intensity;
        const float radius = m->m_Radius;
        float center[2]; Vec2Copy(center, m->m_Center);
        float color[3]; Vec3Copy(color, m->m_Color);

        float xx = (0 <= center[0] - radius) ? center[0] - radius : 0;
        float xxx = (center[0] + radius < g_WindowWidth) ? center[0] + radius : g_WindowWidth - 1;
        float yy = (0 <= center[1] - radius) ? center[1] - radius : 0;
        float yyy = (center[1] + radius < g_WindowHeight) ? center[1] + radius : g_WindowHeight - 1;

        for (int y = yy; y < yyy; y++) {
            for (int x = xx; x < xxx; x++) {
                if ((x - center[0])*(x - center[0]) + (y - center[1])*(y - center[1]) >= radius*radius) {
                    continue;
                }
                float bufferColor[4];
                float p[2] = {x, y};
                FetchRGBAColor(bufferColor, x, y, &g_MetaballBuffer);

/* うまくいかなかった
                float sqrD = Max(g_WindowWidth, g_WindowHeight);
                float nearest[2];
                for (int j = 0; j < g_NumMetaballs; j++) {
                    if (j == i) { continue; }
                    const Metaball2D *m = &g_Metaballs[j];
                    if (sqrD > Vec2SqrDistance(p, m->m_Center)) {
                        sqrD = Vec2SqrDistance(p, m->m_Center);
                        nearest[0] = m->m_Center[0];
                        nearest[1] = m->m_Center[1];
                    }
                }
*/
                bufferColor[3] = bufferColor[3] + CalcMetaballDensity(m, p) * m->m_Intensity;
                bufferColor[0] = (bufferColor[0] + color[0] * bufferColor[3])/2;
                bufferColor[1] = (bufferColor[1] + color[1] * bufferColor[3])/2;
                bufferColor[2] = (bufferColor[2] + color[2] * bufferColor[3])/2;
```

```
/* うまくいかなかった
                if (Vec2SqrDistance(p, center) > Vec2SqrDistance(p, nearest)) {
                    bufferColor[0] = bufferColor[0];
                    bufferColor[1] = bufferColor[1];
                    bufferColor[2] = bufferColor[2];
                } else {
                    bufferColor[3] = bufferColor[3] + CalcMetaballDensity(m, p) * m->m_Intensity;
                    bufferColor[0] = (color[0] * bufferColor[3]);
                    bufferColor[1] = (color[1] * bufferColor[3]);
                    bufferColor[2] = (color[2] * bufferColor[3]);
                }
*/
                SetRGBAColor(bufferColor, x, y, &g_MetaballBuffer);
            }
        }
    }


    /* set up for contouring */
    if (g_DisplayMetaballContours)
        SetupMetaballDensityBuffer();


    /*
     * TODO: apply thresholding.
     */
    for (int yi=0; yi<h; yi++)
    {
        for (int xi=0; xi<w; xi++)
        {
            // normalize the color and density if the density value exceeds g_MetaballThreshold.
            float color[4];
            FetchRGBAColor(color, xi, yi, &g_MetaballBuffer);
            if (color[3] > g_MetaballThreshold) {
                color[0] /= color[3];
                color[1] /= color[3];
                color[2] /= color[3];
                color[3] = 1;
                SetRGBAColor(color, xi, yi, &g_MetaballBuffer);
            } else {
                float zero[4] = {0,0,0,0};
                SetRGBAColor(zero, xi, yi, &g_MetaballBuffer);
            }
        }
    }
}
```

上のようなコードを実行すると次の様になった。



「うまくいかなかった」と書いてある部分のコメントアウトをはずすと次の様になった。