Aufgabe 4

Der z-Buffer-Algorithmus funktioniert immer dann eindeutig, wenn sich die z-Werte der zu zeichnenden Objekte unterscheiden. Sollte jedoch ein Pixel gezeichnet werden und der z-Wert des Objektes, zu dem es gehört, an dieser Stelle gleich dem Wert im z-Buffer sein, so hängt das visuelle Ergebnis von der Zeichenreihenfolge und der z-Buffer-Funktion ab. In diesem Fall sind zwei Hauptfunktionen zu unterscheiden (es gibt wesentlich mehr, diese beiden sind aber die mit Abstand am häufigsten verwendeten):

1. equal-less (`GL_LEQUAL`): Der später gezeichnete Pixel überschreibt den bisherigen im Framebuffer.
2. less (`GL_LESS`): Der alte Pixel bleibt erhalten.

Dreht man die Zeichenreihenfolge der Objekte um, so erhält man das gleiche Ergebnis wie durch Vertauschung der z-Buffer-Funktion.

Ein weiteres Problem liegt in möglichen Rundungsfehlern bei der Diskretisierung der z-Werte, was zu einem sogenannten *z-flickering* führt.

Hinweis: Es gibt noch weitere z-Buffer-Funktionen, die aber nur eine untergeordnete Rolle spielen.

Aufgabe 5

Soll eine Szene mit transparenten Objekten gezeichnet werden, so ist folgende Vorgehensweise empfehlenswert:

1. Schalte Blending aus, aktiviere z-Buffer
2. Zeichne alle lichtundurchlässigen Objekte ($\alpha=1$)
3. Schalte Blending ein, setze Schreibschutz für z-Buffer
4. Zeichne alle transparenten Objekte ($\alpha<1$)

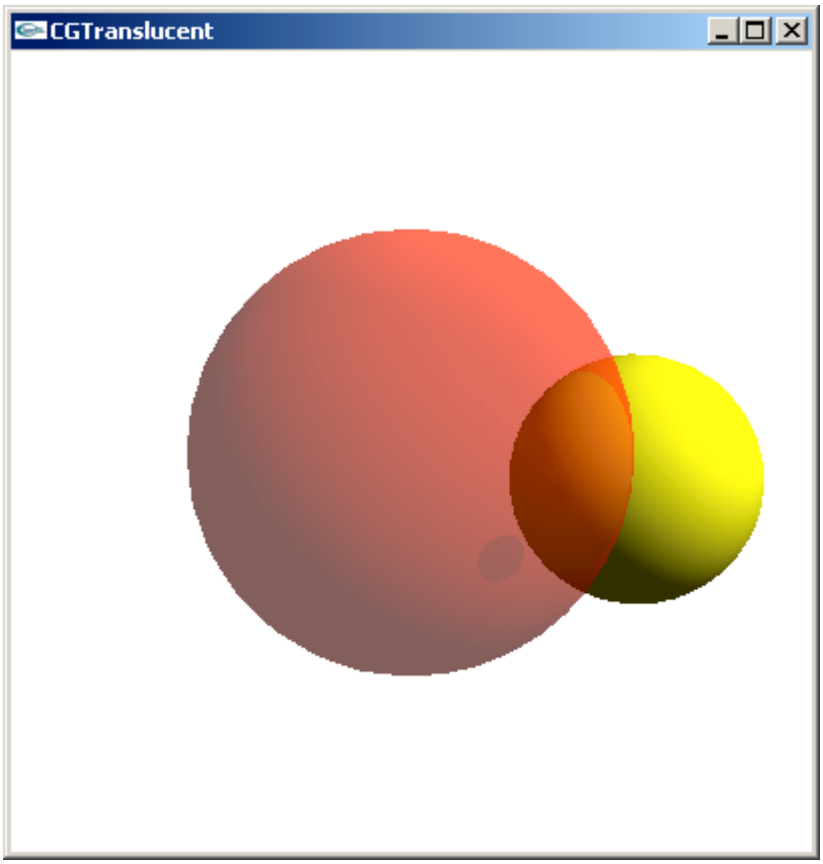
Diese Vorgehensweise ist notwendig, um so eine gewisse Vorsortierung in der Zeichenreihenfolge zu erreichen, da anderenfalls Transparenzen falsch gezeichnet werden. Sobald alle opaken Objekte gerendert wurden, können mit Hilfe der z-Buffer-Werte die transparenten Objekte gezeichnet werden. Wichtig ist, dass die z-Buffer-Werte nicht verändert werden, um auch mehrfache Überlagerungen von Transparenzen zu erlauben.

In OpenGL gestaltet sich die Umsetzung der einzelnen Schritte recht einfach:

1. Blending deaktivieren: `glDisable(GL_BLEND);`
   z-Buffer einschalten: `glDepthMask(GL_TRUE);`
2. Farbwerte entsprechend setzen (α-Anteil immer 1 !) und Objekte zeichnen
3. Blending deaktivieren: `glEnable(GL_BLEND);`
   z-Buffer ausschalten: `glDepthMask(GL_FALSE);`
4. Farbwerte entsprechend setzen (α-Anteil immer kleiner 1 !) und Objekte zeichnen

Für die gegebenen Kugeln entsteht dann der Code:

```
void CGTranslucent::onDraw() {
    // Mechanismus zur Behandlung opaker Objekte
    glDepthMask(GL_TRUE);
    glDisable(GL_BLEND);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glColor4f(1,1,0,1); // nicht Transparent (A=1)
    glPushMatrix();
        glTranslatef(moveX_,0,0);
        glutSolidSphere(0.5, 32, 32);
    glPopMatrix();

    // Mechanismus zur Behandlung transparenter Objekte
    glEnable(GL_BLEND);
    glDepthMask(GL_FALSE);

    glColor4f(1,0,0,0.4); // Transparent (A=0.4)
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glutSolidSphere(0.8, 32, 32);

    // Nicht vergessen! Front- und Back-Buffer tauschen:
    swapBuffers();
}
```



Aufgabe 6

Diese Aufgabe löse ich gemäß Terminverschiebung erst mit dem nächsten Übungsblatt.