

Wintersemester 2010/2011

## Übungen zu Computergraphik I - Blatt 7

Abgabe am 05. 01. 2011

### Aufgabe 1 (BSP-Trees, 12+8 Punkte)

In dieser Aufgabe soll eine Szene mit (semi-)transparenten Objekten korrekt dargestellt werden. Hierfür müssen die Objekte bzgl. Augkoordinaten von hinten nach vorne gerendert werden.

Laden Sie das Projekt `bsp_framework` von der Vorlesungs-Homepage herunter. Über den Menüeintrag *Files*→*Open Model* können Sie Szenen im Wavefront Obj-Format laden. Einige Beispielszenen stehen im Unterverzeichnis `./models` zur Verfügung. Der Obj-Loader kann Szenen aus Vier- und Dreiecken laden, und konvertiert diese in Dreiecke. Für jedes Dreieck sind in der Instanzvariable `m_triangles` die Indices seiner drei Eckpunkte (Vertices) gespeichert. Die dazugehörigen Vertexkoordinaten stehen in der Instanzvariablen `m_vertices`.

- a) In der Klasse `Mesh` wird eine Szenen in oben genannter Form gespeichert. Bauen Sie aus der Menge von Dreiecken einen BSP-Tree auf. Um zu entscheiden, auf welcher Seite der Splitplane ein Dreieck ist, testen Sie der Einfachheit halber einen beliebigen der drei Eckpunkte des Dreiecks, d.h. Splitting ist in dieser Teilaufgabe nicht gefordert.

Implementieren Sie hierzu die bereits vorhandene, aber noch leere, Routine `generateBSP()`. Verwenden Sie die vordefinierte Klasse `Mesh::BspNode`. Die Wurzel des BSP-Baumes soll der Variable `m_bspRoot_splitsDisabled` zugewiesen werden. Die Traversierung inklusive Rendering des BSP-Baumes sind bereits implementiert. Sie können also sofort das Ergebnis Ihrer Implementierung betrachten. Vergessen Sie nicht, den entsprechenden Radiobutton oben zu aktivieren.

- b) Erweitern Sie ihre Methode dahingehend, dass Dreiecke, die die Splitplane schneiden, korrekt unterteilt und die resultierenden Teildreiecke zu dem passenden Kindknoten hinzugefügt werden. Weisen Sie die Wurzel des resultierenden BSP-Baumes der Variablen `m_bspRoot_splitsEnabled` zu. Um zu entscheiden, ob ein Vertex in der Ebene ist oder nicht, müssen Sie Fließkommawerte korrekt vergleichen, d.h.  $|a - b| < \varepsilon$ . Verwenden Sie hierzu die vordefinierte Instanzvariable `Mesh::m_EPS`.

*Tip:* Die folgenden Funktionen der STL-Klassen `std::list` und `std::vector` können für Ihre Implementierung hilfreich sein:

- Um auf Elemente sequentiell zuzugreifen, gibt es die sogenannten "Iteratoren" (eine Art intelligenter Pointer), die wie folgt verwendet werden können:

```
for( std::list<int>::iterator it = liste1.begin(); it != liste1.end(); ++it )
{
    int aktuelle_elem = *it;
}
```

- `begin()` gibt einen Iterator auf das erste Element einer Liste bzw. eines Vectors zurück,
- `end()` das letzte Element, und `size()` die Anzahl der Elementen.
- Ein neues Element kann mit `push_back()` hinzugefügt werden, z.B.

```
std::vector<Triangle> dreiecksliste;  
dreiecksliste.push_back( Triangle( index0, index1, index2 ) );
```