Programmierübung 2: Benutzerinteraktion
Computergrafik, Herbst 2012

Abgabedatum: Donnerstag 18. Oktober, 12:00

Diese Übung muss am Donnerstag 18. Oktober via Ilias abgegeben werden. Zusätzlich muss die Übung wie letztes Mal einem Assistenten am Computer gezeigt werden. Schreiben Sie sich dazu wieder auf der Online Liste ein.

Erstellen Sie für die einzelnen Teilaufgaben separate Eclipse Projekte, damit Sie diese getrennt demonstrieren können. Sie können jeweils das Projekt "simple" kopieren und dieses dann erweitern. Verwenden Sie auch das Zusatzmaterial, das wir in der Datei *Zusatzmaterial.zip* auf Ilias bereitstellen.

1 Kamera und Frustum (2 Punkte)

1.1 Kamera

Erweitern Sie die Klasse *jrtr.Camera* des zur Verfügung gestellten Java Codes so, dass eine Kamera-Matrix aus den 3 Parametern *Center of Projection*, *Look-at-Point* und *Up-Vector* konstruiert werden kann. Die Parameter (Vektoren) sollen als Member-Variablen gespeichert werden und über get/set Methoden zugänglich gemacht werden. Wenn die Parameter verändert werden, muss die Kamera-Matrix jeweils automatisch aktualisiert werden.

1.2 Frustum

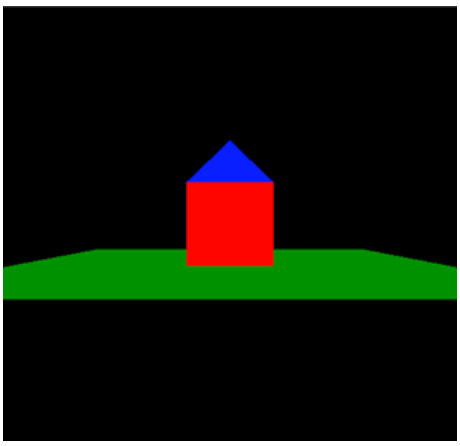
Erweitern Sie die Klasse *jrtr.Frustum* so, dass eine Projektions-Matrix aus den 4 Parametern *Near Plane, Far Plane, Aspect Ratio* und *Vertical Field of View* konstruiert werden kann. Verwenden Sie die Matrix-Formel, die in der Vorlesung gezeigt wurde. Das Verwenden der *glu* Bibliothek oder anderen Bibliotheken zur Berechenung der Kamera- resp. Projektions-Matrix ist dabei nicht gestattet.

1.3 Testen

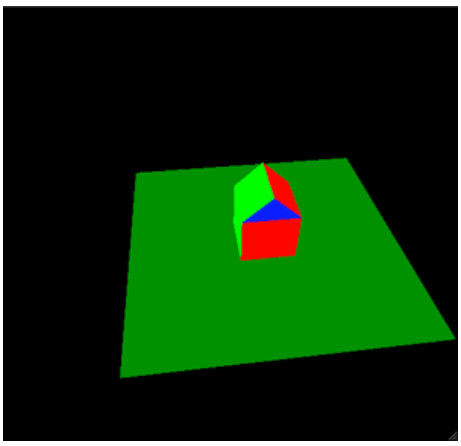
Verwenden Sie die auf Ilias bereitgestellte Test-Szene *house.txt* und rendern Sie zwei Bilder unter Verwendung der folgenden zwei Parameter-Sets, die in der Figur unten gezeigt sind.

Parameter für Bild 1	
Aspect Ratio	1
Vertical Field of View	60 Grad
Near, far clip Planes	1, 100
Center of Projection	0,0,40
Look-at-Point	0,0,0
Up-Vector	0,1,0

Parameter für Bild 2	
Aspect Ratio	1
Vertical Field of View	60 Grad
Near, far clip Planes	1, 100
Center of Projection	-10,40,40
Look-at-Point	-5,0,0
Up-Vector	0,1,0



Ausgabe für Bild 1



Ausgabe für Bild 2

2 Virtueller Trackball (3 Punkte)

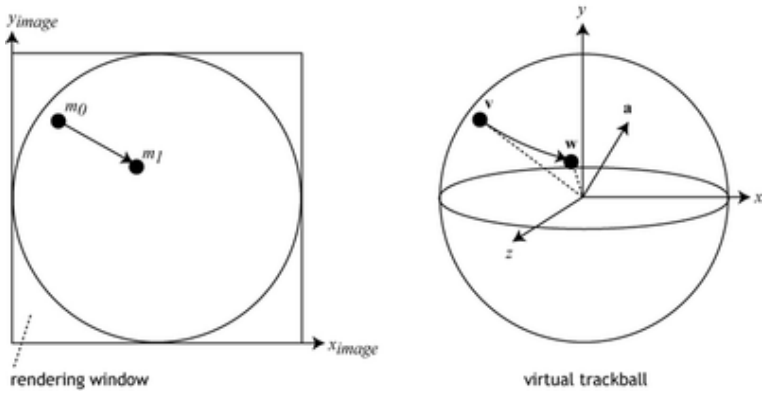
Implementieren Sie einen virtuellen Trackball mit welchem Sie ein Objekt interaktiv mit der Maus rotieren können. Ihre Lösung soll Mausbewegungen bei gedrückter Maustaste (Dragging) in eine Rotationsmatrix umwandeln, welche Sie dann zur Transformation der Szene verwenden. Es sollen Rotationen um alle drei Koordinatenachsen möglich sein.

Die untenstehende Abbildung zeigt wie man aus einer Mausbewegung eine Rotationsachse und einen Rotationswinkel erhält. Wir bezeichnen mit m_0 und m_1 zwei aufeinanderfolgende 2D Mauspositionen. Diese Positionen definieren in 3D zwei Punkte v und w auf einer "virtuellen Kugeloberfläche", die das Rendering Window ausfüllt. Verwenden Sie das Kreuzprodukt $a = v \times w$ als Rotationsachse und den Winkel zwischen v und w als Rotationswinkel.

Horizontale Bewegungen in der Mitte des Fensters sollten zu einer Rotation um die y-Achse führen. Vertikale Bewegungen in der Mitte des Fensters sollten zu einer Rotation um die x-Achse führen. Bewegungen am Rand des Fensters (horizontal und vertikal) sollten zu einer Rotation um die z-Achse führen. Beiliegend auf Ilias finden finden Sie eine detailierte Beschreibung einer Virtual Trackball Implementation.

Behandeln Sie auch folgende Spezialfälle:

- Mausposition liegt ausserhalb des virtuellen Trackballs.
- Das Rendering-Window ist nicht quadratisch.



Visualisierung des virtuellen Trackball.

2.1 Testen

Testen Sie Ihre Implementation mit Dreiecksgittern, die aus Dateien gelesen werden. Wir stellen dazu auf Ilias eine Klasse *ObjReader* zur Verfügung, welche in das *jrtr* Projekt aufgenommen werden soll. Die Klasse liest das *.obj* Dateiformat, ein einfaches text-basiertes Dateiformat zum Speichern von Meshes (Polygonnetzen). Im wesentlichen wird eine Liste mit allen Eckpunkten (Zeilen beginnen mit v, ein Eckpunkt pro Zeile) und eine Liste von Indizes der Eckpunkte aller Polygone (Zeilen beginnen mit f, ein Polygon pro Zeile) gespeichert. Zusätzlich können Normalen (Zeilen beginnen mit vn) und Texturkoordinaten (Zeilen beginnen mit vt) angegeben werden. Weitere Details zum *.obj* Format finden Sie auf Wikipedia und bei der *.obj* Formatspezifikation. Im Zusatzmaterial ist eine Testdatei *teapot.obj* beigelegt.

3 Fraktale Landschaft (3 Punkte)

Implementieren Sie einen Algorithmus, um fraktale Landschaften zu erzeugen. Folgen Sie der Beschreibung von der Übungsstunde, welche auch unter diesem Link detailiert wiedergegeben ist. Ihr Verfahren soll auch Normalenvektoren generieren, indem das Kreuzprodukt von jeweils zwei Dreieckskanten berechnet wird. Wählen Sie die Koordinaten so, dass die xy-Ebene der Grundebene (Boden) und z der Höhe entspricht. Entwickeln Sie ein einfaches Verfahren, um die Farben der Eckpunkte entsprechend ihrer Höhe zu setzen.

4 Interaktive Kamerabewegung (2 Punkte)

Implementieren Sie ein Verfahren, um mit einer Kombination von Tastatur- und Mauseingaben die Kamera interaktiv zu bewegen. Tastatureingaben sollen die Translation der Kamera bestimmen, und die Maus die Rotation. Verwenden Sie zum Beispiel die WASD Tasten für Vorwärts-, Links, Rückwärts-, und Rechtsverschiebungen. Bei gedrückter Taste soll sich die Kamera in die entsprechende Richtung bewegen. Die Maus bestimmt die Rotation der Kamera. Es gibt verschiedene Möglichkeiten, die Rotationsachsen zu wählen, so

dass intuitive Navigation ermöglicht wird. Wir nehmen an, dass die Scene aus einer Welt entsteht, wo die xy-Ebene dem Boden entspricht, und z nach oben zeigt. Wir schlagen vor, dass vertikale Verschiebung der Maus einer Rotation der Kamera um die x-Achse des Kamerakoordinatensystems entspricht. Horizontale Verschiebung der Maus soll der Rotation um die z-Achse des Weltkoordinatensystems entsprechen.

Verwenden Sie die Navigation mit Maus und Tastatur, um das interaktive "Fliegen" durch das Gelände aus Aufgabe 3 zu ermöglichen.