

Computergrafik 1

Abgabetermin:

Die Lösung zu diesem Übungsblatt ist bis zum Freitag den **29. Mai 2009, 12:00 Uhr s.t.** über UniWorx abzugeben.

Inhalt:

Dieses Blatt dient zum Erlernen von grundlegenden Konzepten der Objekt-Selektion und -Manipulation mit Hilfe der Maus. Hierzu werden Sie Ihr bereits erstelltes Fenster erweitern, um Benutzern die Selektion (*Picking*) und Manipulation (hier: *Verschieben*) von Objekten zu ermöglichen. Pro Aufgabe können maximal zehn (Aufgabe 16 maximal 20) Punkte erreicht werden. Zum Bestehen des Übungsblatts müssen in **JEDER** Aufgabe mindestens fünf (Aufgabe 16 mindestens 10) Punkte erreicht werden.

Geben Sie insgesamt (d.h. ein Projekt für alle Aufgaben zusammen) alle benötigten Header-, Source-, und Projektdateien (von *Qt Creator* erzeugt) mit ab, d.h. *.h, *.cpp und *.pro Dateien. Abgaben die nicht kompilieren werden mit 0 Punkten bewertet. Fassen Sie alle Aufgaben zu einer zip-Datei zusammen und laden Sie sie bei UniWorx hoch. Für die Mathematik-Aufgabe (Aufgabe 18) fügen Sie jeweils ein Bild oder ein PDF (falls mit Word oder LaTeX erstellt) in den Ordner „auf3-18“ ein.

Aufgabe 16 Picking mit der Maus

(20 Punkte)

Diese Aufgabe dient zum Erlernen der einfachen Selektion mit Hilfe der Maus. Um solche in *OpenGL* zu realisieren, können Sie die Standardfunktionen `glPickMatrix`, `glLoadName` und `glRenderMode` verwenden. Alternativ dürfen Sie auch mit Hilfe der Funktion `gluUnProject` arbeiten, jedoch ist die hier vorgestellte Vorgehensweise auf den zu Beginn erwähnten Funktionen `glPickMatrix`, `glLoadName` und `glRenderMode` aufgebaut.

- a) Zunächst muss der Modus zum Selektieren durch Drücken der Taste *M* ermöglicht werden. Wird diese Taste gedrückt, wechselt der Modus entweder von *Kamera* auf *Objekt* oder umgekehrt. Die Modi sollen wie folgt arbeiten:
 - *Kamera-Modus*: Dies ist der bisher gewohnte Modus, d.h. Sie können die Kamera verschieben und rotieren. Objekt-Selektionen hingegen sind nicht möglich.
 - *Objekt-Modus*: In diesem Modus bleibt die Kamera an Ihrer Position fest und kann nicht verändert werden. Nun sind Objekt-Selektionen und -Manipulationen (hier *Verschieben*) möglich.

Die nachfolgenden Teilaufgaben beziehen sich **alle** auf den Objekt-Modus. **(4 Punkte)**

- b) Fügen Sie eine Funktion `select` zu Ihrem `GLWidget` hinzu. Diese Funktion erhält als Übergabeparameter die Mauskoordinaten x und y . Setzen Sie nun den Rendermodus mit der Funktion `glRenderMode` auf `GL_SELECT`. Legen Sie sich einen Buffer (d.h. ein Feld vom Typ `GLuint` mit z.B. Länge 512) an, in dem die Ergebnisse gespeichert werden und weisen Sie dies dann als Select-Buffer zu (mit der Funktion `glSelectBuffer`). Initialisieren Sie

anschließend die sogenannten *GL Names* mit der Funktion `glInitNames`. Wechseln Sie nun in den Matrix-Modus der *Projektion* und erstellen Sie sich die Pick-Matrix mit der Funktion `gluPickMatrix`. Wechseln Sie anschließend in den *Modelview*-Modus und führen alle Transformationen durch, die Sie auch beim Zeichnen der Szene durchführen (hier: Platzieren der Szene relativ zur Kamera). Während dem Rendern der Szene müssen Sie nun für jedes Objekt einen „Namen“, geben (**Hinweis:** Dies ist ein Ganzzahlwert, den Sie später benötigen, um das getroffene Objekt identifizieren zu können.). Um diesen Namen zu setzen, rufen Sie **vor** dem Rendern des Objekts die Funktion `glLoadName` auf und übergeben ihr ein Integer. (**Tipp:** Sie können hier den Index den das Objekt in der Liste innerhalb der Szene besitzt verwenden, dann ist die Identifikation später sehr einfach). (6 Punkte)

- c) Achten Sie ferner darauf, dass das Grid bei einem Rendervorgang mit `GL_SELECT` **nicht** mitgezeichnet wird, da es nicht selektierbar sein soll. Sie können dafür die Übergabe-Parameter der Funktion `render` anpassen, d.h. übergeben Sie dieser Funktion z.B. ein Wert vom Typ `bool`, der aussagt, ob gerade im Selektions-Modus gezeichnet wird. (2 Punkte)
- d) Wenn Sie die Szene nun gerendert haben (im Modus `GL_SELECT`), können Sie sich mit Hilfe der Funktion `glRenderMode` und dem Parameter `GL_RENDER` die Anzahl der „getroffenen“ Objekte speichern. Zusätzlich liegen im zuvor angelegten Buffer weitere Informationen zu den selektierten Objekten (je 4 pro Objekt). Für das erste Objekt gilt demnach:
- `buffer[0]`: Anzahl der *Names*, die auf dem Stack zum Zeitpunkt des Treffers lagen.
 - `buffer[1]`: Minimaler *z*-Abstand des Objekts zum Zeitpunkt des Treffers (d.h. die Vorderseite aus Sicht des Benutzers).
 - `buffer[2]`: Maximaler *z*-Abstand des Objekts zum Zeitpunkt des Treffers (d.h. die Rückseite aus Sicht des Benutzers).
 - `buffer[3]`: Der Name des Objekts, das getroffen wurde (d.h. der *Name*, der vorher von Ihnen vergeben wurde - also der Index in der Liste aller Objekte).

Da bei einem Selektionsvorgang häufig mehrere Objekte (d.h. `hit > 1`) selektiert werden (wenn sie z.B. hintereinander liegen) können Sie nun mit Hilfe des minimalen *z*-Abstands herausfinden, welches Objekt das zum Benutzer am nächsten liegende ist. Dieses Objekt soll dann selektiert werden, d.h. setzen Sie einen Wert in dem betreffenden `SceneObject`. Lassen Sie das selektierte Objekt nun noch in weiß zeichnen, um den Benutzer adäquates Feedback zu geben. (6 Punkte)

- e) Wenn der Benutzer ein anderes Objekt selektiert, kein Objekt selektiert (d.h. in den leeren Raum klickt, also `hit = 0`) oder das bereits selektierte Objekt nochmals anklickt, soll das zuvor selektierte Objekt wieder deselektiert werden. (2 Punktw)

Aufgabe 17 Verschieben der Objekte

(10 Punkte)

In dieser Aufgabe sollen Sie nun die zuvor selektierten Objekte verschieben können. Dazu werden Sie die Objekte zunächst mit Achsen versehen, die dann als Verschiebe-Hilfen benutzt werden.

- a) Geben Sie jedem `SceneObject` eine Funktion `renderAxis`, die zwei Parameter erhält. Der erste Parameter bezeichnet dabei die Achse (0 für die *x*-Achse, 1 für die *y*-Achse und 2 für die *z*-Achse), der zweite Parameter zeigt an, ob gerade im Modus `GL_SELECT` gerendert wird. Dies ist wichtig für eine spätere Teilaufgabe. (1 Punkt)

- b) Zeichnen Sie jede Achse in ihrer typischen Farbe, d.h. die x -Achse in *rot*, die y -Achse in *grün* und die z -Achse in *blau*. Eine Achse besteht aus einem *Quader* von bestimmter Länge, sowie einer *Pyramide*, die die Spitze symbolisieren soll. Modifizieren Sie die Szene derart, dass die Achsen immer über allen Objekten (auch über dem Grid) gezeichnet werden (**Tipp:** Die Funktion `glDisable` lässt Sie z.B. die Tiefenüberprüfung deaktivieren.). (2 Punkte)
- c) Im Selektionsmodus können Objekte anders gerendert werden. Dies ist dann sinnvoll, wenn ein Objekt sehr klein ist, der Selektionsradius jedoch etwas höher ausfallen soll, um dem Benutzer die Auswahl zu vereinfachen. Daher sollten die Achsen im Modus `GL_SELECT` etwas größer gezeichnet werden. **Tipp:** Im Selektionsmodus ist es ausreichend einen *Quader* zu zeichnen, der sowohl den ursprünglichen *Quader* als auch die ursprüngliche *Pyramide* umgibt. (1 Punkt)
- d) Um die Zugehörigkeit der Achsen zu einem Objekt zu gewährleisten, können Sie die Objekt-Ids (d.h. die *Names* die Sie zuvor vergeben haben) jeweils mit 4 multiplizieren. Die Achsen liegen dann jeweils dazwischen. Dies bedeutet, dass das Objekt n den Namen $4 \times n$ erhält und die zugehörigen Achsen mit $4 \times n + 1$ (x -Achse), $4 \times n + 2$ (y -Achse) und $4 \times n + 3$ (z -Achse) bezeichnet werden. (2 Punkte).
- e) Beim Testen auf einen Treffer in der Szene sollen die Achsen vorrangig betrachtet werden, d.h. selbst wenn die Achse hinter einem Objekt liegt (auch wenn Sie davor gezeichnet wird) soll diese selektiert werden. Modifizieren Sie daher Ihren Code für die Überprüfung auf das am nächsten liegende Objekt dementsprechend. (1 Punkt).
- f) Nun sollen die Objekte verschiebbar sein. Wird also bei einem `mousePressEvent` eine Achse getroffen, soll bei einem `mouseMoveEvent` das Objekt entlang dieser verschoben werden. Bedienen Sie sich hierbei der Projektion der Fensterkoordinaten in Weltkoordinaten, die Sie bereits beim Verschieben der Kamera verwendet haben. Alternativ dürfen Sie auch `gluUnProject` verwenden. (3 Punkte).

Aufgabe 18 Bézier-Kurven

(10 Punkte)

Bézier-Kurven sind ein hilfreiches Werkzeug in 2D- und 3D-Computergrafik. In dieser Aufgabe erlernen Sie den Umgang mit solchen Kurven, die Sie in einer späteren Übung auch programmatisch umsetzen werden. Bézierkurven n -ten Grades sind wie folgt definiert:

$$C(t) = \sum_{i=0}^n B_{i,n}(t) P_i \text{ wobei } B_{i,n}(t) = \binom{n}{i} \cdot t^i (1-t)^{n-i} \text{ für } t \in [0, 1] \text{ Beispiel für Grad 1:}$$

$$C(t) = \sum_{i=0}^1 t^i (1-t)^{1-i} P_i = (1-t)P_0 + tP_1 \text{ für } t \in [0, 1]$$

- a) Erstellen Sie die Formel für eine Kurve vierten Grades. (2 Punkte)
- b) Wie viele Kontrollpunkte enthält eine Kurve n -ten Grades? (1 Punkt)
- c) Gegeben sei eine Kurve dritten Grades, die durch folgende Punkte definiert ist: $P_0 = (-3, -2)^T$, $P_1 = (-1, -3)^T$, $P_2 = (1, 3)^T$ und $P_3 = (3, -11)^T$. Berechnen Sie die Punkte **und** die Steigung der Kurve an den Stellen $t_1 = \frac{1}{3}$, $t_2 = \frac{1}{2}$ und $t_3 = \frac{2}{3}$. (4 Punkte).
- d) Kurven zweiten Grades können unter gewissen Voraussetzungen in Funktionen umgewandelt werden. Welche Voraussetzung muss erfüllt sein, damit eine Kurve zweiten Grades umgewandelt werden kann? Begründen Sie Ihre Antwort und skizzieren Sie ein Beispiel für eine Kurve, die umgewandelt werden kann, sowie eines für eine Kurve, die nicht umgewandelt werden kann. (3 Punkte)