OpenGL

Matthias Voigt

Einführung

OpenGL ist die Kurzform von Open Graphics Library, welches von Silicon Graphics um 1992 entwickelt wurde. Die Grafikbibliothek ist grundsätzlich plattformunabhängig und eine Softwareschnittstelle zur Grafikhardware.

Zum Grundverständnis von OpenGL ist es wichtig, dass OpenGL ein Zustandsautomat ist – damit bleiben einmal gesetzte Zustände erhalten bis sie geändert werden. Weiterhin werden eigene Datenstrukturen definiert sowie die verschieden Prozeduren – also hauptsächlich Funktionen ohne Rückgabewerte. Damit ist der OpenGL-Code sehr leicht in verschiedenen Programmiersprachen zu implementieren, da die Syntax recht ähnlich ist.

In Delphi benutzt man OpenGL sehr einfach: man downloadet einfach eine Header Datei, kopiert sie ins Programm-Verzeichnis oder in einen anderen für Delphi bekannten Ordner. Danach wird die Unit einfach im uses Block eingebunden. Zwei empfehlenswerte Header sind dglOpengL und openGL12. Die meisten Unterschiede zwischen den verschiedenen Implementierungen sind die Art der Initialisierung und der Anfangszustand der Maschine. Als fortgeschrittener Programmierer stört dies natürlich nicht, da man eh jeden Zustand nach eigenen Vorstellungen setzen sollte, um Komplikationen zu vermeiden.

Funktionen sind meist nach folgendem Schema benannt:

glFunktionXY().

Zuerst steht die Bibliothek (meist gl, glu oder glut), danach der eigentliche Funktionszweck, und danach Anzahl und Art der Parameter.

Die Ausgabe von Objekten kann in mehreren Arten geschehen. Die wichtigsten Primitiven sind hierbei

- GL_POINTS einfache Pixelwolken
- GL_LINES 2 Punkte ergeben eine Linie
- GL_TRIANGLES 3 Punkte ergeben immer ein Dreieck
- GL_TRIANGLE_STRIP es entstehen (Anzahl der Punkte)-2 Dreiecke
-

Ausgabefunktionen müssen dabei immer zwischen folgendem Block stehen, wobei X für eine der vorhergehenden Konstanten steht:

```
glBegin(X);
  // Ausgabe
glEnd;
```

Die Koordinaten werden durch ein kartesisches Koordinatensystem angegeben, bei dem die x-Achse am unteren Rand von links nach rechts liegt. Die y-Achse ist am linken Rand von unten nach oben und die z-Achse geht vom Bildschirm zum Betrachter.

OpenGL arbeitet intern mit 3 Matrixstapeln welche jeweils ein Koordinatensystem beschreiben. Sie dienen zur Steuerung der Objekte, Steuerung der Kamera und Steuerung der Texturen auf den Oberflächen. Der Grundzustand ist jeweils die Einheitsmatrix.

Zur einfachen Erstellung von Objekten bearbeitet man die Modellmatrix, in welcher man die Primitiven hinzufügt. Man kann das Koordinatensystem auch verändern und neu erstellte Vertexes werden dann auf der neuen abgebildet.
Zur Veränderung benutzt man grundsätzlich die Matrizenmultiplikation, welche nicht kommutativ ist. Zum Beispiel ist die Reihenfolge Verschiebung → Drehen verschieden von Drehen → Verschiebung, wie man einfach überlegen kann.

OpenGL arbeitet mit einer Transformationspipline. Nach Eingabe der Vertexes erfolgt die Modelltransformation in das Koordinatensystem. Weiter werden die Koordinaten je nach Kameraposition transfortmiert und die orthogonale oder perspektivische Transformation durchgeführt, wobei auch das Clipping dazugehört. Nach der perspektivischen Division erfolgt nur noch die Viewport-Transformation für die Ausgabe im Monitor.

Transformation

Durch Modellmatrix-Transformationen können die Objekte wie gewünscht im Raum platziert werden. Dazu gibt es die Translation, die Skalierung (mit dem Sonderfall der Spiegelung) , die Rotation und die Scherung.

Translation

Die Verschiebung erfolgt durch den Translationsvektor t. Dies ist eine einfache Addition der Vektoren v und t, wobei v der Objektvektor ist. In Matrizenschreibweise ist dies die 4te Spalte.

Skalierung

Die Skalierung geschieht durch den Skalierungsvektor s und erfolgt immer vom Ursprung aus. In Matrizenschreibweise ist dies hier die Diagonale.

Spiegelung

Die Spiegelung ist eine Sonderform der Skalierung. Dazu werden einfach negative Skalierungswerte genommen.

Rotation

Die Rotation um einen Winkel r erfolgt immer um eine bestimmte Achse. Bei einer Rotation um eine Koordinatenachse berechnen sich die Matrizenwerte einfach aus dem Additionstheorem. Eine einfache Eingabe erfolgt durch glRotate.

Scherung

Die Scherung ist eine Verschiebung eines Punktes parallel zu den Ebenen. Hierfür bietet OpenGL keine Implementierung und muss selbst erarbeitet werden.

Ansichtstransformation

Eine Szene in OpenGL kann wie in der realen Welt aus verschiedenen Positionen betrachtet werden. Dazu nutzt man die Ansichtstransformation. Hier gibt es den Augpunkt, also die Position der Kamera, den Blickbezugspunkt, welches die Position beschreibt, wohin man schaut und den Aufwärtsvektor, der schließlich beschreibt, welche Richtung auf dem Monitor oben ist. Die Berechnung vor der Ausgabe erfolgt durch Multiplikation der Modellkoordinaten mit der Ansichtstranformationsmatrix. In OpenGL setzt man diese durch gluLookAt mit den entsprechenden Parametern.

Perspektivische Transformation

Um die drei-dimensionalen Punkte auf einer zwei-dimensionalen Fläche abzubilden, wird die perspektivische Transformation als Vorstufe der perspektivischen Projektion gebraucht. Nach dem Strahlensatz kann man die X und Y Werte einfach durch Z dividieren und erhält dann die 2D Werte. Aber dabei hat man das Problem, dass keine Tiefe mehr gespeichert wird, die Ausgabe weiß also nicht, welches Pixel gezeichnet werden muss. Außerdem kann man dann kein Clipping an Deckflächen realisieren, sondern nur an den Mantelflächen – damit wird weit weg, und zu nah dran bzw. hinter einem ignoriert.

Orthogonale Transformation

Der Grundsatz ist ähnlich der perspektivischen Transformation. Hier liegt allerdings das Projektions- Zentrum im Unendlichen hinter dem Betrachter. Die Projektionsstrahlen sind nun also parallel und aus dem Sichtvolumen wird ein Quader.

Clipping

Um die Geschwindigkeit zu steigern werden Punkte außerhalb des Sichtvolumens nicht gezeichnet. Dies geschieht wie bereits im Vortrag über Clipping beschrieben.

Hierarchische Transformation

Bei Verwendung einer einzelnen Matrix hätte man schon bei 2 unabhängigen Objekten Probleme. Daher wäre es sinnvoll bei neuen Opjekten immer vom gleichen Bezugspunkt aus zu erstellen. Dafür bietet Opengl 3 Matrix-Stacks, für jedes Koordinatensystem einen.

Angenommen, man will ein Sonnensystem modellieren, müsste man ohne den Matrixstack die Erd- und Mondkoordinaten entweder kompliziert errechnen oder alle Transformationen durch inverse Matrizenberechnungen rückgängig machen. Dies ist sehr aufwendig und dazu bei riesigen Szenen stark rechenlastig und dazu unnötig. Denn bei OpenGL speichert man eine Matrix im Stack und kann diese später bei Bedarf wieder abrufen.