Praxisorientierte Einführung in die Computergraphik

- Vorlesungsplan -

1. Einführung

In dieser Lektion erhalten die Teilnehmer einen Überblick über das Wissensgebiet der Computergraphik und ihrer Anwendungsgebiete. Dabei werden zunächst grundlegende Begriffe wie Graphik, Computergraphik und Rendern erläutert. Ein geschichtlicher Abriss beginnend mit dem Jahr 1950 bis zur Gegenwart zeigt anhand wichtiger Meilensteine die Entwicklung der Computergraphik hin zu einer eigenständigen Disziplin der Informatik.

Die Vorlesung wird ins Teilgebiet der generativen Computergraphik eingeordnet und die Berührungspunkte zu den Wissenschaftsgebieten Bildanalyse und Bildverarbeitung werden betrachtet.

Im letzten Teil der Vorlesung wird auf Notwendigkeit, Inhalte und Anwendungen wichtiger Standards der Computergraphik eingegangen. Dabei wird unter anderem die historische Entwicklung des OpenGL-Standards vorgestellt.

2. Grundkonzepte der OpenGL

Der Industriestandard OpenGL wird als technische und methodische Grundlage des Kurses motiviert. Gemeinsamkeiten, aber auch Unterschiede zu anderen Standards werden aufgezeigt. Dabei wird den Teilnehmern anhand von Beispielen erläutert, dass die in der Vorlesung vermittelten Algorithmen und Verfahren grundlegend für alle aktuellen Standards zum Rendern von dreidimensionalen Szenen sind, d.h. auch für DirectX gelten.

Der Einstieg in OpenGL wird Top-Down gewählt. Beginnend mit der allgemeinen Struktur des Verarbeitungsschemas der OpenGL wird der zentrale Apparat der Renderpipeline erläutert. Die Definition wichtiger Begriffe wie Kontext, GL Status, Shader leitet die ausführliche Diskussion über Aufgaben und Struktur der einzelnen Stufen der GL-Pipeline ein. Das Konzept des Zustandsautomaten wird eingeführt und damit die quasi-sequenzielle Verarbeitung von 3D-Geometriedaten bis hin zu eingefärbten Pixeln im Framebuffer aus globaler Sicht erläutert.

3. Das Templateprogramm

OpenGL bietet keine Unterstützung zur Umsetzung des Userinterfaces. Beispielsweise stehen keine Funktionen zum Handling von Fenstern, Dialogelementen oder zum Abfragen von Tastatur oder Maus zur Verfügung. Vielmehr liegen diese Aufgaben in der Verantwortung des Nutzers. Je nach Betriebs- und Fenstersystem unterscheiden sich die Herangehensweisen zur Programmierung eines Rahmensystems entsprechend deutlich. Das soll aber nicht Inhalt dieses Kurses sein.

Aus diesem Grund steht für die eigenen Experimente der Teilnehmer mit OpenGL ein Templateprogramm zum Download bereit, das genau diese Aufgaben übernimmt. Dessen Struktur und Funktionsweise wird in dieser Lektion vorgestellt. Alle weiteren Lektionen und die Übungen basieren auf diesem Template. Neben dem Einbinden von GL-Code zum Zeichnen von Geometrie werden die Programmierung von Tastatur- und Mausinteraktion oder Ansatzpunkte für das Umsetzen von Animationen, z.B. Zählvariablen oder der Zugriff auf Framecounter erläutert.

4. Zeichenprimitive der OpenGL

Jede graphische Szene basiert auf grundlegenden Beschreibungselementen. Für OpenGL Anwendungen sind das die so genannten Vertex-Daten. Aufbauend auf den bereits eingeführten Grundkonzepten wird die Rolle des Vertex als zentrales Datum in OpenGL erläutert.

Beginnend mit den beiden Konzepten zum Handling von Vertex-Daten (Begin/End-Block und Vertex Buffer) werden die zehn standardisierten Block-Primitive zur Spezifikation von Punkten, Linien und polygonalen Flächen im Raum detailliert erläutert. Dabei wird mit Programmbeispielen auf die vielfältigen Möglichkeiten eingegangen, die OpenGL zum Zeichnen von Szenen bietet. Die Beispiele verdeutlichen anhand von Vertex-Attributen wie Farbwerten ebenfalls das Konzept des Zustandsautomaten, das der GL-Pipeline zugrunde liegt.

Im Anschluss an die Diskussion der einfachen Flächenprimitive (Dreiecke, Vierecke) wird das GL-Polygon eingeführt. Dabei werden die limitierenden Bedingungen, die OpenGL an die Spezifikation von Polygonen stellt, aus Sicht eines einfachen und gut in Hardware umsetzbaren Pipeline-Designs nachvollziehbar begründet.

Geometrische Grundkörper wie Kugel, Würfel und Kegel und gekrümmte Flächen sind nicht im GL-Kern standardisiert, werden aber von den Utility-Bibliotheken bereitgestellt. Die Diskussion dieser Zeichenprimitive (Quadrics, NURBs) zielt auf die Möglichkeiten zum schnellen Prototyping graphischer Szenen ab.

5. Verbundprimitive, Arrays und Buffer

Aufbauend auf den GL-Grundprimitiven motivieren Effizienzbetrachtungen die Verwendung so genannter Verbundprimitive zum Zeichnen komplexer polygonaler Oberflächen. Anwendung und Leistungsfähigkeit, aber auch Einschränkungen der standardisierten Fan- und Strip-Primitive werden vorgestellt.

Verbundprimitive stellen aber nur eine Art zum effizienten Hochladen von Geometriedaten an die GL-Pipeline dar. Die andere, noch effizientere Methode stellen so genannte Vertex Arrays oder Vertex Buffer dar. Die GL unterscheidet Vertex Arrays (VA) und Vertex Buffer Objects (VBO). Die Lektion macht die Teilnehmer mit den Grundkonzepten und der programmtechnischen Anwendung von VA und VBO vertraut.

6. Transformationen

Transformationen bilden die neben Zeichenprimitiven und Attributen die wichtigste Grundlage für die Umsetzung graphischer Modelle. In dieser Lektion wird die Transformationskette der OpenGL erläutert. Ausgehend von mathematischen Grundlagen wie affine Transformationen und praktischen Anwendungsszenarien werden die Transformationsschritte MODELVIEW, PROJEKTION und VIEWPORT vorgestellt und deren Programmierung erläutert. Ein wichtiger Aspekt ist dabei die Platzierung und Positionierung geometrischer Bestandteile in der 3D-Szene durch die Manipulation des GL-Koordinatensystems (Koordinatentransformationen). Der zweite Aspekt betrifft die Umsetzung eines Kameramodells (Perspektive, Projektion).

Neben diesen Grundfunktionen zum Setzen oder Beeinflussen der im Server State gespeicherten Transformationsmatrix für Geometrie und Kamera wird insbesondere der Matrix-Stack als ein Werkzeug zum Beherrschen von komplexen Transformationsketten (Transformationshistorie) vorgestellt. Schließlich werden am Template die Techniken für Kamerafahrten und Animationen in Form von akkumulierten Transformationen demonstriert.

7. Szenengraph

Die Teilnehmer sind mit dem Wissen aus den Lektionen 1-6 prinzipiell in der Lage, eine 3D OpenGL-Anwendung zu schreiben. Damit größere Modelle bzw. Programme beherrschbar sind, bedarf es geeigneter Konzepte. Das Mittel zum Zweck sind im Allgemeinen Szenengraphen. OpenGL bietet dafür jedoch keine Funktionalität.

Da die geeignete Strukturierung großer Modelle aber essenziell für eine erfolgreiche programmtechnische Umsetzung mit OpenGL ist, werden die Teilnehmer in dieser Lektion mit den Grundkonzepten des Szenengraphen vertraut gemacht. Begriffe wie Wurzelknoten, Strukturen, Lokalität und Vererbung werden eingeführt und anhand von Beispielen sinnvolle Ansätze für Szenengraphen vermittelt. Besonderes Augenmerk wird

der Strukturierung der geometrischen Transformationen in der Szene zuteil sowie deren funktionale Umsetzung mit den OpenGL-Transformationskonzepten und dem Matrix-Stack.

Unabhängig von OpenGL wird das Szenengraph-Konzept auch unter Gesichtspunkten wie verteilte Implementation, Effizienzbetrachtungen und kommerzielle Bibliotheken betrachtet.

8. Graphische Austauschformate

OpenGL stellt eine vielseitige und performante Renderingschnittstelle zur graphischen Hardware dar. Die vorangegangenen Lektionen haben die grundlegenden Konzepte zum Rendern dreidimensionaler Geometrie vermittelt. OpenGL bietet aber keinerlei Modellierungsfunktionalität.

Anspruchsvolle, detailreiche und animierte 3D-Szenen werden im Normalfall mit spezialisierten Programmsystemen -den so genannten Modellern- gestaltet. Mit Hilfe von Austauschformaten lassen sich die so erstellten graphischen Modelle zwischen Programmsystemen austauschen und in selbst erstellte Programme importieren.

In dieser Lektion wird das Problem des Austauschs graphischer Daten zunächst grundsätzlich diskutiert, auch unter den Gesichtspunkten von Inkompatibilitäten bestimmter Modellierungstechniken und proprietärer Dateiformate.

Im zweiten Teil der Lektion wird das universelle Austauschformat Wavefront (.obj) vorgestellt. Nach einer Erläuterung der Ausdruckskraft und der Syntax folgen der Entwurf geeigneter Datenstrukturen sowie einer Importfunktion für Wavefront-Files, deren Inhalt dann mit den Möglichkeiten der OpenGL dargestellt wird. Die Programmierung der Renderfunktion erfolgt zur Vertiefung der Kenntnisse aus den Lektionen 3-5 sowohl mit einfachen GL-Primitiven, mit Arrays und auch mit dem Buffer-Konzept (VBO).

9. Beleuchtung mit OpenGL

Die Computergraphik bedient sich zur Steigerung der Realitätsnähe der dargestellten Abbilder vor allem Beleuchtungsmodellen und Texturierungsverfahren. In dieser Lektion werden grundlegende Aspekte der Beleuchtungsberechnung vorgestellt. Zunächst werden globale und lokale Beleuchtungsmodelle gegenübergestellt. Dabei wird sowohl auf die Möglichkeiten (Spiegelungen, Schatten, phys. Korrektheit) als auch auf die softwareseitige Umsetzung (Algorithmen, Anforderungen, Laufzeiten) eingegangen. Beispielhafte Abbildungen stehen zur Verfügung.

Im zweiten Teil wird das lokale Beleuchtungsmodell von OpenGL (per vertex lighting) vorgestellt und dessen Anwendung umfassend erläutert. Ein wichtiger Bestandteil zur

Berechnung der Lichtsituation an Vertices ist der Normalenvektor. Seine Bestimmung wird für die verschiedenen Zeichenprimitive und für den Wavefront-Lader erläutert.

10. Texturierung

Texturen sind aus aktuellen Graphikprogrammen nicht wegzudenken. Neben der Beleuchtung verleihen ihre vielfältigen Anwendungsmöglichkeiten im Renderprozess den berechneten Bildern die gewünschte Realitätsnähe.

OpenGL bietet mit aktuellen Shaderkonzepten und Extensions eine umfassende Unterstützung fortgeschrittener Texturierungsverfahren inkl. Multitexturing. Diese Konzepte zu behandeln würde den Umfang des Kurses sprengen. Aus diesem Grund wird exemplarisch die grundlegende Technik der Anwendung von Bildtexturen vermittelt und mit praktischen Beispielen in Kombination mit dem OpenGL Beleuchtungsverfahren demonstriert.