Prüfungsprotokoll

Graphische Systeme und Betriebssysteme

Prof. Heinrich Müller

21. Oktober 2002

Note: 1.0

Graphische Systeme

Prüfer: In der Computergraphik sind ja Flächen sehr beliebt und da gibt es ja diese B-Spline-Flächen. Erklären Sie die mal.
Ich: Im Gegensatz zu den B-Spline-Kurven wird bei den B-Spline-Flächen nicht nur einen Kontrollpolygonzug von de-Boor-Punkten sondern ein Kontrollnetz von $(k+1)\cdot(l+1)$ de-Boor-Punkten benutzt (siehe Abbildung 1). Desweiteren sind ein zweidimensionaler Parameterbereich $[0, u_{k+m+1}[\times[0, v_{l+n+1}[\subset \mathbb{R}^2$ sowie zwei Knotenvektoren $(u_0, u_1, \ldots, u_{k+m+1})$ und $(v_0, v_1, \ldots, v_{l+n+1})$ gegeben.

Soll nun der Flächenpunkt zu einem Punkt (u, v) aus diesem Parameterbereich berechnet werden, so wird zuerst entlang jeder der $k+1$ Zeilen ein neuer Punkt zum Parameterwert v ausgerechnet. Dabei wird jede Zeile als eine B-Spline-Kurve vom Grad n mit $l+1$ Kontrollpunkten interpretiert. Daraus ergeben sich also $k+1$ neue Punkte, welche als Kontrollpunkte einer B-Spline-Kurve vom Grad m mit $k+1$ de-Boor-Punkten interpretiert werden. Für

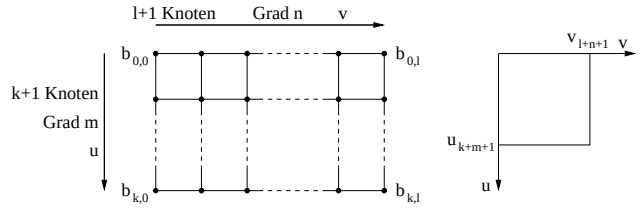
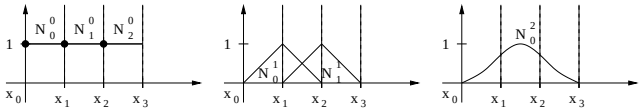


Abbildung 1: Kontrollnetz und Parameterbereich einer B-Spline-Fläche



(a) B-Splines $N_i^0(x)$ (b) B-Splines $N_i^1(x)$ (c) B-Spline $N_0^2(x)$

Abbildung 2: Aussehen der B-Splines $N_i^0(x)$, $N_i^1(x)$ und $N_i^2(x)$ für einen Knotenvektor $\xi = (x_0, x_1, x_2, x_3)$

diese Kurve wird zum Parameterwert u ein neuer Punkt berechnet, welcher der gesuchte Flächenpunkt zu (u, v) ist. Als Formel hingeschrieben sieht dies wie folgt aus:

$$b(u,v) = \sum_{i=0}^{k} \left(\sum_{j=0}^{l} d_{i,j} \cdot N_j^n(v) \right) \cdot N_i^m(u)$$

Prüfer: Jetzt haben sie das ja schon gleich so schön hingeschrieben. Aber wo gehen in dieser Formel denn die Knotenvektoren ein?
Ich: In den B-Spline-Funktionen $N_j^n(v)$ und $N_i^m(u)$.

Prüfer: Und wie sehen diese B-Spline-Funktionen grob aus?
Ich: Die B-Spline-Funktionen sind rekursiv definiert, wobei sich eine B-Spline-Funktion vom Grad n ergibt, indem an zwei B-Spline-Funktion vom Grad $n-1$ eine steigende bzw. fallende lineare Funktion ranmultipliziert wird. Die Funktionen sehen dann in etwa so aus, wie in Abbildung 2.

Prüfer: Angenommen ich habe eine B-Spline-Fläche bzw. allgemein eine Fläche in Parameterdarstellung gegeben und möchte diese durch ein Dreiecksnetz darstellen. Wie könnte ich das machen? Mir reicht eine einfache Lösung.
Ich: Dann könnte man einfach den Parameterbereich gleichmäßig unterteilen, die entsprechenden Flächenpunkte ausrechnen und als Eckpunkte der Dreiecke benutzen.

Prüfer: Wie sieht denn die Parameterdarstellung einer Fläche allgemein aus?
Ich: Bei einer Fläche in Parameterdarstellung im $\mathbb{R}^3$ ist ein zweidimensionaler Parameterbereich $P \subset \mathbb{R}^2$ gegeben. Die Koordinaten des Ergebnispunktes zu einem Parameterwert $(u, v) \in P$ ergeben sich dann wie folgt:

$$k(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix},\ x,y,z : \mathbb{R}^2 \to \mathbb{R},\ (u,v) \in P$$

Prüfer: Wie würde ich das machen, wenn ich nun ein Dreiecksnetz mit OpenGL darstellen möchte?
Ich: Da OpenGL in Form einer Zustandsmaschine implementiert ist, würden zuerst einmal die Attribute wie Strickdicke und Farben eingestellt werden, mit denen die darauf folgenden Primitive später alle gezeichnet werden. Anschließend würde mit Hilfe des Befehls

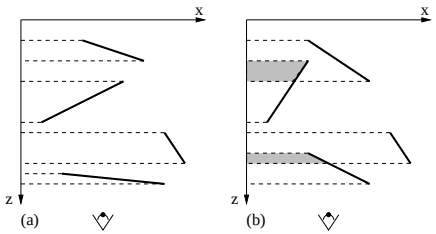


Abbildung 3: Problem der überlappenden z-Bereiche beim Painters Algorithmus

`glBegin( GLEnum mode )` das Zeichnen einleitet werden. Der günstigste Modus zum Zeichnen eines Dreiecksnetzes wäre wahrscheinlich `GL_TRIANGLE_STRIP`. Dann werden die einzelnen Punkte per `glVertex()` an OpenGL übergeben und nachdem dies geschehen ist, wird das Ganze per `glEnd()` beendet.

Prüfer: Warum würden Sie es als `GL_TRIANGLE_STRIP` an OpenGL übergeben?
Ich: Da bei diesem Modus weniger Punkte angegeben werden müssen. Ansonsten würden einige Punkte mehrfach angegeben werden, was Bandbreite verschwendet.

Prüfer: Ok. Ein weiteres Problem in der Computergraphik ist ja die Entfernung verdeckter Flächen. Welche Algorithmen kennen Sie da?
Ich: Ich kenne die folgenden Verfahren: Brute Force, Tiefenpuffer, Scan-Line, Bereichsunterteilungsverfahren nach Warnock und die Prioritätslistenverfahren wie den Painters Algorithmus.

Prüfer: Wie funktioniert denn der Painters Algorithmus? Was ist denn dort das Grundverfahren?
Ich: Beim Painters Algorithmus werden die Elemente der Szene nach fallender Entfernung zum Betrachter hin sortiert und dann in dieser Reihenfolge nacheinander gezeichnet. Dabei kann die Sortierung z.B. nach dem kleinsten z-Wert einer jeden Fläche geschehen. Das Überdeckungsproblem wird dadurch gelöst, dass weiter entfernt liegende Flächen im Bildwiederholspeicher von näher liegenden Flächen überzeichnet werden können.

Prüfer: Das klappt aber nicht immer.
Ich: Stimmt. Wenn sich die z-Bereiche einiger Flächen überlappen (siehe Abbildung 3), dann müssen weitere Tests durchgeführt werden. Habe die vier Tests schnell skizziert. Herr Müller wollte wissen, warum die Tests in dieser Reihenfolge durchgeführt werden, worauf ich geantwortet hab, dass diese nach Effizienz sortiert sind.

Prüfer: Welche Verfahren zur Beleuchtungsberechnung oder Schattierung kennen sie?
Ich: Radiosity und Raytracing. Anschließend anhand einer Skizze Raytracing erklärt. Auf Schattenberechnung und Spiegelung eingegangen. Brechung wollte er schon nicht mehr hören. Erwähnt dass ein primitiver Ansatz sehr ineffizient sein kann.

Prüfer: Sie haben ja schon gesagt, dass das nicht so effizient ist. Wie gehts besser?
Ich: Gesagt, dass zwei Ziele verfolgt werden: Schnitttests mit Objekten sollen schneller gehen und die Anzahl der Schnitttests soll minimiert werden. Auf Hüllquader allgemein eingagangen und geschachtelte Hüllquader erklärt.

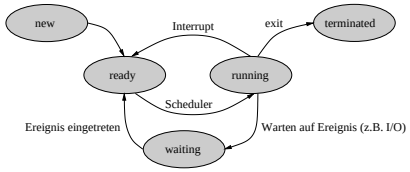


Abbildung 4: Die verschiedenen Zustände eines Prozesses

Betriebssysteme

Prüfer: Ok, kommen wir zum Thema Betriebssysteme. Eine grundlegende Einheit ist ja ein Prozess. Was ist das?
Ich: Kurz gesagt, dass ein Prozess ein Programm in der Ausführung ist. Einige Daten, die im Process Control Block (PCB) stehen, erwähnt.

Prüfer: Es gibt ja mehrere Zustände, in denen ein Prozess sein kann. Welche sind das?
Ich: Abbildung 4 hingemalt. Gesagt, dass es „new“ in modernen Systemen wahrscheinlich gar nicht mehr gibt. Dann wurde es etwas verwirrend bezüglich des Pfeils, der vom „running“ nach „ready“ geht (Interrupt). Herr Müller hatte sich irgendwas überlegt, wo ich bis heute nicht weiß, was er hören wollte. Ich hab noch irgendwas von CPU-Protection gefaselt.

Prüfer: Ok, Prozesse benutzen ja verschiedene Resourcen, um die sie konkurrieren. Dabei kann es ja zu Deadlocks kommen. Wie sind diese charakterisiert?
Ich: Eine Menge von Prozessen befindet sich in einem Deadlock-Zustand, wenn jeder Prozess der Menge auf ein Ereignis wartet, welches nur durch einen anderen Prozess in dieser Menge ausgelöst werden kann. Als Beispiel vier wartende Autos an einer Rechts-vor-Links-Kreuzung hingezeichnet. Anhand dieser die vier Bedingungen (Mutual Exclusion, Hold and Wait, No Preemption, Circular Wait) erklärt.

Prüfer: Angenommen, von jeder Resource gibt es nur eine Instanz. Wie kann man dann einen Deadlock erkennen?
Ich: Als Beispiel einen entsprechenden Resource Allocation Graph (RAG) gezeichnet und auch erklärt, was ein RAG ist. Herr Müller wollte z.B. wissen, was die Pfeile bedeuten (Request und Assignment Edges). Erklärt, dass ein Deadlock vorliegt, wenn der RAG einen Kreis enthält.

Prüfer: Wie kann man denn feststellen, ob ein Graph einen Kreis enthält?
Ich: Mit Hilfe einer Tiefensuche (DFS) auf dem Graph in Laufzeit $O(|V| + |E|)$.

Prüfer: Wenn die Prozesse auf geteilte Resourcen zugreifen, müssen die sich ja auch teilweise koordinieren.
Ich: Dazu benutzt man die Verfahren der Prozess-Synchronisation.

Prüfer: Und was kennen sie da?
Ich: Software-Lösungen, wie z.B. Bankers Algorithmus oder Lösungen für zwei Prozesse.

Prüfer: Bankers Algorithmus?
Ich: Ach nein, ich meine natürlich den Bakery Algorithmus.

Prüfer: Wofür ist denn der Bankers Algorithmus?

Ich: Der Bankers Algorithmus zur Behandlung von Deadlocks, wenn man Deadlock Prevention betreibt. War nun bezüglich der Unterschieds zwischen Prevention und Avoidance etwas verwirrt und musste den Unterschied erklären.

Prüfer: Ok, kommen wir jetzt aber zur Prozess-Synchronisation zurück. Was gibt es noch?
Ich: Semaphoren und Monitore?

Prüfer: Und wie funktioniert so ein Monitor?
Ich: Zeichnung aus dem Buch skizziert. Erläutert, dass die geteilten Daten nur über die Methoden manipuliert werden können und dass Monitore ein Programmierkonstrukt sind. Gesagt, dass sich immer nur ein Prozess im Monitor befinden kann und die Queue erwähnt. Das reichte anscheinend auch schon, die Condition Variablen musste ich nicht mehr erklären.

Prüfer: Und woran erinnert ein Monitor programmiertechnisch?
Ich: An Konzepte der Objektorientierung.

Prüfer: Ja, das sind Objekte. Und was ist eine Semaphore?
Ich: Die Definition mit Busy-Wait hingeschrieben.

Prüfer: Und ohne Busy-Wait?
Ich: Die Definition ohne Busy-Wait hingeschrieben und die entsprechenden System Calls erwähnt.

Prüfer: Gibt es irgendeinen grundlegenden Unterschied zwischen zählenden und binären Semaphoren?
Ich: Nein. Er wollte hören, dass zählende Semaphoren auch durch binäre Semaphoren realisiert werden können. `P()` und `V()` hingeschrieben und dabei `count`, `countLock` sowie `semLock` erklärt.

Kommentar: Herr Müller ist als Prüfer wirklich sehr zu empfehlen. Die Prüfungsatmosphäre ist sehr ruhig und entspannt. Dazu hat wahrscheinlich auch der Umstand beigetragen, dass Herr Müller nicht sofort stur mit der Prüfung angefangen hat, sondern mich erst einmal gefragt hat, was ich denn bis zum Zeitpunkt der Prüfung so im Studium gemacht habe und sich mit mir z.B. kurz über das Thema meiner Projektgruppe unterhalten hat. Dadurch kam man langsam ins Gespräch und ich wurde auch ruhiger.

Das Lernen für diese Prüfung hat sich bei mir etwas in die Länge gezogen, insgesamt dürften es jedoch wohl so ca. drei Monate gewesen sein. Die meiste Zeit wird dabei wahrscheinlich für die Erstellung von eigenen Zusammenfassungen mit LaTeX draufgegangen sein, was ich jedoch nur weiterempfehlen kann. Wenn man den Stoff einmal in eigenen Worten hinschreibt, merkt man nämlich recht schnell, ob man wirklich alles verstanden hat und weiß auch schon in etwa, wie man etwas in der Prüfung erklären kann.

Wie man wahrscheinlich merkt, wollte ich das Protokoll am Anfang erst etwas ausführlicher schreiben, was ich dann nachher jedoch gelassen habe, da es zu zeitaufwändig wurde. Genauere Antworten finden sich unter anderem in den oben schon erwähnten Zusammenfassungen zu den beiden Prüfungsgebieten, welche unter der folgenden Adresse zu finden sind:

http://www.michaelgregorius.de/

Viel Glück für die Prüfung!

Literatur

[Fellner 1992] Fellner, Wolf Dietrich (1992). *Computergrafik*. BI Wissenschaftsverlag, Zweite Aufl.

[Foley 1994] Foley, James D. (1994). *Grundlagen der Computergrafik*. Addison-Wesley, Erste Aufl.

[Müller 2000] Müller, Heinrich (2000). *Skript Graphische Systeme*.

[Silberschatz 1998] Silberschatz, Abraham (1998). *Operating System Concepts*. Addison-Wesley, Fünfte Aufl.

[Silberschatz 2000] Silberschatz, Abraham (2000). *Applied Operating System Concepts*. Wiley, Erste Aufl.

[Tanenbaum 1995] Tanenbaum, Andrew S. (1995). *Moderne Betriebssysteme*. Hanser, Zweite Aufl.