Prüfungsprotokoll Vertiefungsfach

Prüfer:	Prof. Dr. L. Kobbelt
Beisitzer:	Dipl. Inform. David Bommes
Inhalt:	Computergrafik I+II Geometrische Modellierung I+II
Datum:	1. Oktober 2007
Dauer:	45 Minuten
Vorbereitungszeit:	Drei Monate
Note:	1.0

1 Bemerkungen

Zunächst einmal das Übliche: das Protokoll erhebt natürlich keinen Anspruch auf Vollständigkeit und Korrektheit. Das gesamte Gespräch ist sinngemäß wiedergegeben. Obwohl ich versucht habe, möglichst viele Details aus der Prüfung niederzuschreiben, ist es möglich, dass ich die ein oder andere Frage vergessen habe. Ich hoffe trotzdem, dass Dir das Protokoll einen guten Eindruck vom Ablauf der Prüfung vermittelt.

Die Atmosphäre in der Prüfung war sehr angenehm und ruhig, was mir bei meiner eigenen Nervosität sehr entgegen kam. Prof. Kobbelt lässt einen stets ausreden und falls man mal nicht weiter weiß, führt er einen an die Lösung heran. Falls man es dann immer noch nicht weiß, erklärt er es am Ende selbst. Ich glaube sogar, dass ihm etwas daran liegt, dass man in der Prüfung noch etwas lernt. Es ist übrigens nicht schlimm, wenn man mal ein paar Formeln verdreht. Das Verständnis ist da deutlich wichtiger. Die Prüfung begann mit 15 Minuten Verspätung, was bei drei Monaten Vorbereitungszeit aber auch keinen Unterschied mehr macht.

Falls Du Fragen zum Protokoll hast oder Fehler findest, schreib mir einfach eine eMail: malte.weiss(at)rwth-aachen.de

2 Vorbereitung

Für die Prüfung in der Computergrafik-Vertiefung sollte man sich schon genug Zeit nehmen. Entweder lernt man es richtig oder man lässt es gleich. Prof. Kobbelt will die Dinge schon genau wissen und fragt manchmal im Detail nach. Außerdem gibt es gewisse Beweise, die man einfach können muss. Ich denke zwar, dass Prof. Kobbelt ein paar Lieblingsthemen hat, allerdings kann letztendlich alles dran kommen (es sind auch schon so exotische Themen für Differentielle Geometrie geprüft worden). Wer die Vorlesung/Übungen gehört (und aufgepasst) hat, ist allerdings klar im Vorteil!

Computergrafik I kann man nahezu vollständig aus dem Skript lernen, wobei ein Blick auf die Foliensätze nicht schaden kann, um gewisse Zusammenhänge besser zu verstehen. Die Vorlesungsvideos sind natürlich optimal, um letzte Unklarheiten zu beseitigen (z.B. die Herleitung der 1/2/3-Punkt-Konstruktionen). Transformationen sind ein Standard-Thema in der Prüfung und sollten im Detail beherrscht werden! Ich denke, Volume Rendering und Shadows sind die weiteren Schwerpunkte. Weiterhin sollte man wissen, wie man Shadow Volumes auf der GPU berechnet (steht nicht im Skript).

Computergrafik II Endlich gibt es auch dazu ein Skript. Ich habe weitestgehend daraus gelernt, auch wenn es zum Schluss etwas schwächer wird. Ich habe nur wenige zusätzliche Papers gelesen, aber im Teil Global Illumination kommt man nicht drum herum.

Geometrische Modellierung I Die eigene Mitschrift und die angegebene Literatur (Prautzsch und Farin) waren hier meine Lerngrundlage. Farin ist etwas anschaulicher, Prautzsch in manchen Gebieten aber näher an der Vorlesung.

Geometrische Modellierung II Die Vorlesung konnte ich aufgrund eines Auslandaufenthalts leider nicht hören, was sich als echtes Problem darstellte. Ich lernte aus mindestens fünf unterschiedlichen Mitschriften von Kommilitonen. Prautzsch und Farin halfen auch, die Mitschriften waren aber wichtiger.

3 Prüfungsfragen

Nun zum interessanten Teil ...

3.1 Computergrafik I

In CGI kam ich erstmal nicht wirklich in die Gänge. Zwar konnte ich zu allem etwas sagen, aber die Fragen zu den Transformationen, die zeitlich mindestens die Hälfte dieses Blocks einnahmen, führten bei mir zu der ein oder anderen Verwirrung. Das fiel aber offenbar nicht zu stark ins Gewicht. Schwerpunkte: Transformationen und Shadows.

- **Fangen wir mit CG1 an. Eines der wichtigen Konzepte sind ja die homogenen Koordinaten. Erklären Sie das mal.**
  (x, y, z) äquivalent zu $(\omega x, \omega y, \omega z, \omega)$. Dehomogenisierung durch Projektion auf $\omega = 1$ Ebene, usw.

- **Was ist denn ein Fluchtpunkt?**
  Habe die Definition genannt, erklärte dann, wie man einen solchen Fluchtpunkt per Konstruktion findet (Verschieben einer Gerade in das Projektionszentrum und Schnittpunkt mit der Bildebene). Den mathematischen Beweis (Standardprojektion

von $L(\lambda) = a + \lambda u$ für $\lim \lambda \to \infty$) wollte Prof. Kobbelt dann aber nicht mehr vollständig hören.

- **Jetzt schreiben Sie doch mal eine der perspektivischen Transformationen auf.**
  Eigentlich eine Standardfrage. Habe die Standard-Projektion und dann die nächst allgemeinere Form (Betrachter im Zentrum, Bildebene bei n mit Fokaldistanz δ) aufgeschrieben. Unglücklicherweise wollte ich glänzen und behauptete, dass wir eine Parallelprojektion erhalten, wenn δ gegen unendlich konvergiert. Das trifft aber leider nur für die dritte allgemeine Projektion zu (Bildebene im Zentrum, Betrachter bei $-n$, Fokaldistanz δ).

- **Wir haben ja über die Konstruktion von 1-/2-/3-Punkt-Perspektiven gesprochen. Was bedeutet denn die Zahl?**
  Anzahl der Achsen, die nicht parallel zur Bildebene sind.

- **Was für eine k-Punkt-Perspektive ist denn die Standard-Projektion?**
  Da habe ich lange rumgedruckst, bis ich dann durch kleine Hilfestellung auf die Antwort „1-Punkt-Perspektive“ kam.

- **Und die von ihnen genannte nächst allgemeinere Form?**
  Äh ... nach Anfertigung einer kleinen Skizze ... 3-Punkt-Perspektive.

- **Okay, dann haben wir über Shadows gesprochen. Welche Verfahren gibt es denn da?**
  Projective Geometry, Shadow Textures, Shadow Volumes, Shadow Maps, und als Verbesserung der Shadow Maps - noch die Projected Shadow Maps.

- **Welches von den Verfahren ist denn am besten?**
  Hier muss man abwägen. Ich bevorzugte zunächst Shadow Volumes und Shadow Maps, da sie heute auf der GPU berechnet werden können, und nannte ihre Vor- und Nachteile.

- **Jetzt erklären Sie bitte kurz in jeweils zwei Sätzen Shadow Volumes und Shadow Maps.**
  Es waren dann doch noch ein paar mehr Sätze ...

- **Welches von den beiden ist denn jetzt besser?**
  Das hängt von der Situation ab.

- **Dann nennen Sie doch mal eine Situation, wo sich Shadow Maps nicht einsetzen lassen.**
  Die Kerze auf einem Tisch ist da ein gutes Beispiel (wie schön, dass es Prüfungsprotokolle gibt). Da die Kerze uniform in alle Richtungen strahlt, müsste man die Shadow Map mindestens sechs Mal rendern.

- **Wie ist denn die Silhouette eines Objekts definiert?**
  Was folgte war meine improvisierte und wahrscheinlich umständlichste Art, Silhouetten zu definieren :) Prof. Kobbelt sagte darauf, dass die meisten Studenten eine

einfachere Definition nennen würden: Die Kanten zwischen Front- und Backfacing-Faces.

3.2 Computergrafik II

Computergrafik II war von der Themenwahl eher so, wie ich es aus Prüfungsprotokollen kannte, allerdings wurde das Wissen nicht einfach abgefragt, sondern sollte eher angewendet werden (Transfer).

- **Wir haben über Laser Scanning gesprochen. Wie funktioniert denn ein Laser Scanner?**
  Ich erläuterte es so, wie es im Skript stand: Dreieck aus Laser, Kamera und Reflektionspunkt ergibt Tiefenwert. Man erhält ein Range Image, das man dann trianguliert.

- **Welche Probleme treten denn im Allgemeinen beim Scanning auf?**
  Verdeckungen und spekulare Reflektionen. Das Verdeckungsproblem kann man abschwächen, indem man Kamera und Laser näher aneinander stellt, dann wird das Verfahren aber numerisch instabil (ungünstige Dreiecksform).

- **Sinngemäß: Wie werden denn die Daten dann weiterverarbeitet?**
  Ich erläuterte zunächst Registrierung, also *Iterative Closest Points* und brachte noch *Normal Piercing* ein. Prof. Kobbelt war aber am meisten an der Absoluten Orientierung interessiert, die ich dann umfassend erklärte, inkl. der Herleitung der Quaternionen-Rotationen. Leicht verrannt habe ich mich am Ende nur bei der Behauptung, die Least Squares-Matrix hätte drei Eigenvektoren. Irgendwann wurde mir aber klar, dass es vier sein müssen (4×4-Matrix).

- **Wenn wir so einen Datensatz haben, dezimieren wir die Meshes in der Regel anschließend. Welche Verfahren haben wir denn da kennengelernt?**
  Vertex Clustering, Inkrementelle Dezimierung, Remeshing, das sich noch in isotrope and anisotrope Verfahren unterteilt.

- **Welches Verfahren finden Sie denn davon am besten?**
  Inkrementelle Dezimierung, weil wir Zielkomplexität und globalen Fehler einstellen können.

- **Welche Nachteile hat denn Vertex Clustering?**
  Oversampling von uniformen Bereichen, Undersampling von Features, Zielkomplexität nicht einstellbar ...

- **Wieso könnte Vertex Clustering denn trotzdem gut für Laser Scans sein?**
  Äh ...

- **Nun, welche Nachteile hat Vertex Clustering denn noch?**
  Ich grübelte ein wenig. ... Ach so. Na ja, es kann keine Two-Manifold-Meshes garantieren?

- **Inwiefern könnte dieser Nachteil denn gut für Laser Scans sein?**
  Weiter gegrübelt ... „Ganz ehrlich. Das weiß ich nicht."

- **Na ja, Vertex Clustering erzeugt nicht zwangsläufig Two-Manifold-Meshes, aber es *erwartet* sie auch nicht. (In der Nachbesprechung erläuterte Prof. Kobbelt diesen Zusammenhang noch einmal im Detail. Tja, wieder was gelernt.)**

- **Dann erklären Sie mal Inkrementelles Dezimieren.**
  Das habe ich dann getan.

- **Wonach entscheiden wir denn nun konkret, welche Operationen die höchste Priorität kriegen?**
  Anhand von Fehlermaßen, wir haben globale und lokale kennengelernt.

- **(Prof. Kobbelt nickt.) Dann mal ganz konkret bei Half-Edge-Collapse.**
  Na ja, mit Fehlerquadriken.

- **Wie funktionieren die denn?**
  Dann habe ich Fehlerquadriken hergeleitet und wie man einen Lösungsvektor am einfachsten findet ($Ax_{opt} = -t$).

- **Warum benutzen wir denn gerade die Fehlerquadriken?**
  Sie stellen einen globaler Fehler dar, den wir lokal aktualisieren können.

- **Nun erklären Sie mal, wie das konrekt bei einem Half-Edge-Collapse funktioniert.**
  Das habe ich dann getan (Fehlerberechnung mit Hilfe der Summe von Quadriken usw.). Ich erinnere mich, dass ich mich hier mal bei einer Formel verschrieben habe, war aber offenbar nicht schlimm. Zum Schluss sagte ich, dass Fehlerquadriken aber auch nicht optimal seien.

- **Wieso?**
  Na ja, wir zählen Ebenen doppelt und berechnen stets nur den minimalen Least Squares-Abstand zu Ebenen und nicht zu Dreiecken.

- **Nun haben wir folgende Situation. Nehmen wir an, wir haben ein Dreiecksnetz mit exakt einer Millionen Dreiecke, und wir wollen nun für Vertex-Decimation die Quadriken speichern. Wieviel Speicher benötigen wir denn da?**
  Ich berechnete eine Anzahl von Bytes, wobei ich die Euler-Formel zu Hilfe nahm.

- **Okay, wie kann man das denn noch optimieren?**
  Wir müssen nicht unbedingt alle Quadriken speichern, da ja nicht alle Half-Edge-Collapse relevant sind.

- **Doch, die sollten wir schon speichern. Wie können wir den Speicher denn noch reduzieren?**
  Äh ... also gut. Dann packte ich mal die Entropy-Codierung aus.

- **Sie denken viel zu kompliziert. Denken Sie mal einfacher. Ich will herausfinden, ob Sie auch einfache Dinge wissen. (grinst)**
  Äh ... öh ... Valenz-Codierung ...

- **Also, die Quadriken sind symmetrisch. Da können wir doch Speicher einsparen.**
  Ach soooo. Na klar!

- **Wir haben dann die Rendering-Equation kennengelernt. Schreiben Sie die doch mal auf.**
  Ich schrieb die Hemispheren-Form auf und benannte die darin vorkommenden Therme. Die Surface-Schreibweise wollte er nicht mehr sehen.

- **Was bedeutet denn das $\cos\theta$ in der Gleichung?**
  Der Winkel zwischen der Normale an der Position x und ω'.

- **Können Sie das mal an einer Zeichnung visualisieren?**
  Das tat ich dann auch.

- **Wie lösen wir denn die Rendering-Equation?**
  Das ist analytisch nicht möglich. Wir haben finite Elemente und Monte Carlo-Methoden kennengelernt.

- **Erklären Sie mal die Monte Carlo-Methoden.**
  Ich leitete zunächst die Monte Carlo Integration her, erwähnte noch die (eher schlechte) Konvergenz von $O(\frac{1}{\sqrt{N}})$ und das Importance Sampling.

- **Also, wie lösen Sie denn nun die Rendering-Equation?**
  Mit dem Monte Carlo Path Tracing-Algorithmus. Den erklärte ich dann bis ins Detail. Hier sollte man auf jeden Fall auch wissen, wie das Gewicht in jedem Schritt aktualisiert wird und was die einzelnen Terme bedeuten. Prof. Kobbelt wollte insbesondere auch wissen, wie wir den Algorithmus vereinfacht haben (anhand der Verteilungsfunktion von $p(\omega')$, so dass sich alle Terme wegkürzen und das Gewicht jedesmal nur mit 1 multipliziert wird).

3.3 Geometrische Modellierung I

- **Okay, dann kommen wir zu Geometrische Modellierung I. Was ist denn eine Spline?**
  Ein Element des Spline-Raums (nein, das war nicht als Witz gemeint). Und der ist so definiert ...

- **Warum ist das denn überhaupt ein Raum?**
  Äh ... also ... öh ... ich kann jedes Polynom vom Grad n damit darstellen, aber nicht umgekehrt ...

- **Hmm ... was ist denn ein Vektorraum?**
  Ich muss bei der Frage ziemlich große Augen gemacht haben. In meinem Kopf

lief die Festplatte an, die den Stoff von Linearer Algebra I aus dem 1. Semester enthielt. Ich brachte noch halbwegs die Definition raus (Tripel aus Menge von Vektoren und zwei Operatoren $+$ und $\cdot$, Additivität, Homogenität usw.)

- **(Prof. Kobbelt grinst.) Immer wieder interessant, welche Antwort man auf diese Frage bekommt.**
  Letztendlich ging es darum: Der Spline-Raum ist natürlich deswegen ein Vektorraum, weil wir die darin enthaltenen Splines addieren und skalieren können und sie dann immer noch im Spline-Raum liegen.

- **Welche Dimension hat denn der Spline-Raum?**
  Ich sagte $m + n$, doch das passte nicht zu meiner Spline-Raum-Definition, die Splines über die Knoten u_0 bis u_m beschrieb. Nach ein wenig hin und her, korrigierte ich dann meinen Spline-Raum, so dass er über den Knotenvektor u_{-n} bis u_m definiert war. Dann hat's gepasst.

- **Nun hatten wir ja eine Basis für diesen Raum gefunden ...**
  Ja, die B-Splines.

- **Wie sind die definiert?**
  Ich wollte alle drei Definitionen (Truncated Powers, Rekursion und Faltung für uniforme Knotenvektoren) nennen, wurde aber nach erster bereits unterbrochen.

- **Erklären Sie bitte mal die Terme in der Truncated Power-Definition.**
  Truncated Powers und Divided Differences erläutert. Bei den Divided Differences habe ich auch noch die Interpretation genannt (führender Koeffizient des Polynoms, das die gegebene Funktion an den gegebenen Knotenvektoren interpoliert).

- **Ja, aber nun haben die B-Splines ja vier praktische Eigenschaften, die wir in dieser Definition wieder finden.**
  Die habe ich dann auch benannt. $(-1)^{n+1}$ für Positivität, $(u_{i+n+1} - u_i)$ für Partition of Unity, $[u_i ... u_{n+i+1}]$ für den kompakten Support und $(t-u)_+^n$ für π_n, C^{n-1}.

- **Wieso sichert uns diese Definition denn den Support $[u_i ... u_{n+i+1})$?**
  Die Funktion ist links von u_i Null aufgrund der Truncated Powers ... und, na ja, rechts von u_{i+n+1} wegen der Divided Differences.

- **Ja, dass mit den Divided Differences haben Sie gerade schon gesagt. Aber warum ist das denn so? Können Sie das etwas anschaulicher erklären?**
  Ich kam echt nicht drauf. Die korrekte Antwort, die er dann selbst nannte, hatte ich offenbar zuvor schon angedeutet. Es liegt irgendwie daran, dass der führende Koeffizient des Polynoms, dass die Knoten $\{u_i, ..., u_{i+n+2}\}$ interpoliert, Null ist (ohne Gewähr). Zum Verständnis ist es hier offenbar hilfreich, sich das Ganze mal aufzumalen.

- **Wofür brauchen wir überhaupt die Partition of Unity?**
  Damit wir die B-Spline-Kurven in der Form $s(t) = \sum_i c_i N_i^n(t)$ schreiben können.

- **Wie werten wir denn B-Splines aus?**
  Mit dem de Boor-Algorithmus. Der resultiert direkt aus der ... äh ... Mansfield-Cox-Boehm-Rekursion.

- **Echt? Wusste gar nicht, dass Boehm auch mit von der Partie war.**
  Äh ... nein. Mansfield-Cox-de Boor. Nachdem wir alle erstmal herzhaft gelacht haben, beschrieb ich den Algorithmus im Detail anhand einer Zeichnung. Ich verwendete die übliche B-Spline vom Grad $n = 3$ und wertete es in einem allgemeinen Intervall zwischen u_i und u_{i+1} aus. Ich bezeichnete die involvierten Kontrollpunkte mittels globaler Indizierung mit c_{i-3} bis c_i.

- **Wie kommen Sie denn genau auf diese Kontrollpunkte?**
  Das sind die Kontrollpunkte, deren Basisfunktionen das auszuwertende Intervall $[u_i, u_{i+1})$ überlappen.

3.4 Geometrische Modellierung II

- **Gut, Geometrische Modellierung II. Wir haben über Box Splines gesprochen. Wie sind die denn definiert?**
  Ich habe zwei Definition aufgeschrieben: die „Schatten"-Definition, wobei ich eine allgemeine lineare Abbildung nahm (keine Orthogonalprojektion), und die Faltungsdefinition. Bei der Faltungsdefinition betonte ich nochmal explizit, dass das Interval $[0, 1)^s$ halboffen sein muss (im Term $x \in [v_1...v_s][0, 1)^s$). Prof. Kobbelt fragte später noch nach einer Begründung (ich rechtfertigte das mit der Partition of Unity).

- **Welchen Grad haben Box-Splines?**
  $n - s$.

- **Und welche Stetigkeit?**
  $k - 2$, wobei k die minimale Anzahl der Basisvektoren ist, die wir aus der Matrix $[v_1...v_n]$ entfernen müssen, damit sich ihr Rang reduziert (also ganz nach Definition).

- **Nennen Sie mal eine Box-Spline vom Grad 2, der C^1 stetig ist.**
  $M_{1111} = \begin{pmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$

- **Zeichnen Sie bitte mal den Support der Funktion auf?**
  Ich nannte ihn zunächst formal und begann dann, ihn aufzuzeichnen. Echt interessant, wie doof man sich dabei anstellen kann ...

- **Nennen Sie mal eine Box-Spline vom Grad 3, der C^2 stetig ist.**
  $\begin{pmatrix} 1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 1 & 1 & 2 \end{pmatrix}$
  Was man sich hier merken kann: Eine Box-Spline vom Grad n mit Stetigkeit C^{n-1} besteht (zumindest für $s = 2$) immer aus paarweise linear unabhängigen Vektoren.

- **Was ist denn nun schlecht an Ihrer Spline?**
  Der Support ist ungünstig ... und liefert „unschöne“ Polynomstücke. Ich habe mich da noch ein wenig verwirrender ausgedrückt.

- **Wie sind denn die Polynomstücke einer Box-Spline definiert?**
  Da zitierte ich den Satz aus dem Prautzsch-Buch.

- **Wie funktioniert Subdivision?**
  Wollen Sie die CG- oder die GM-Antwort?

- **Na ja, wir sind hier bei Box-Splines.**
  Okay, wir stellen eine Box-Spline als Summe translierter und skalierter Box-Splines dar. Ich formalisierte den Zusammenhang sehr stark, also mit Two-Scale-Relation, Definition von $\alpha_j^{[v_1 \dots v_n]}$ usw. Außerdem ging ich darauf ein, dass wir die Subdivison-Matrix leicht erzeugen können und neben der Gewichtung der „Unter“-Box-Splines auch vier Subdivision-Regeln ableiten können.

- **Dann nennen Sie mir die Subdivision-Matrix für ihre erste Box-Spline (M_{1111}).**
  $$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \to \tfrac{1}{2} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 0 \end{pmatrix} \to \tfrac{1}{4} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$
  Ich benannte ferner die sich ergebenen Subdivision-Regeln.

- **Okay, dann nehmen wir jetzt mal so 3×3-Gitter aus Rechtecken. Führen Sie doch bitte die Subdivision auf diesem Gitter aus.**
  Als Prof. Kobbelt diese Frage stellte, wurde mir zum ersten Mal klar, wie die Subdivision von Box-Splines *wirklich* funktionierte. Erfreulicherweise genau im richtigen Moment. Etwas zögerlich, aber korrekt führte ich dann die Subdivision durch.

- **Dann gehen Sie bitte kurz nach draußen.**

Nach wenigen Minuten wurde ich wieder reingeholt und erfuhr meine Note.