Computergrafik SS 2012

Probeklausur 1

06.07.2012

Universität Osnabrück
Henning Wenke, M. Sc.
Sascha Kolodzey, B. Sc., Nico Marniok, B. Sc.

Aufgabe 1 (19 Punkte)

Beantworten Sie die folgenden Fragen prägnant.

Wie berechnet sich die Länge eines Vektors $v = (x, y, z)$?

Welche Koordinaten hat der Vertex $(4, 2, 6, 2)$ im R^3?

Geben Sie einen zu $v = (1,2)$ orthogonalen Vektor an.

Geben Sie einen dunkelgrünen Farbton in RGB an.

Zeichnen Sie ein konkaves Polygon.

Nennen Sie eine feste Stage der OpenGL Graphics Pipeline.

In welchem Koordinatensystem wird Clipping in OpenGL ausgeführt?

Nennen Sie einen Nachteil der lokalen Beleuchtung.

Nennen Sie einen Vorteil der lokalen Beleuchtung.

Welche Indizes bilden Dreiecke unter der Topologie GL_TRIANGLE_STRIP und den Indexdaten (0, 1, 2, 3, 4)? Achten Sie auch auf konsistente Orientierung.

Wie viele Indizes braucht man bei GL_TRIANGLE_STRIP mindestens um *n* Dreiecke zu erzeugen?

Was bewirkt `barrier(CLK_LOCAL_MEM_FENCE)` in einem Kernel?

Was liefert die GLSL Methode `reflect` und was für Parameter bekommt sie?

Wie oft wird der Vertexshader in einem Durchlauf der Graphics Pipeline ausgeführt?

Wie viele Fragments können pro Pixel minimal und maximal erzeugt werden?

Geben die Formel zur Winkelberechnung zweier Vektoren an, die das Kreuzprodukt verwendet.

Welche Bedingung muss gelten, damit die Multiplikation $M_1 \cdot M_2$ mit $M_1 \in R^{n \times m}$ und $M_2 \in R^{q \times p}$ möglich ist?

Wodurch wird der Öffnungswinkel der Kamera eines Raytracers bestimmt?

Aufgabe 2 (1 + 3 = 4 Punkte)

Gegeben seien die beiden Vektoren $v_1 = (3,0,4)$ und $v_2 = (6,8,0)$. Berechnen Sie den Kosinus des Schnittwinkels α der beiden Vektoren. Geben Sie dabei Ihre Ausgangsformel und den Rechenweg an.

Ausgangsformel:

Berechnung:

Aufgabe 3 (1 + 1 + 3 + 5 = 10 Punkte)

Stellen Sie jeweils die 3×3-Matrix auf, die durch die folgenden Transformationen beschrieben wird.

- $S\left(1, \frac{1}{2}, 2\right)$, eine Skalierung mit den Faktoren $1, \frac{1}{2}$ und 2, jeweils in x-, y- und z-Richtung.
- $T(-1, -2, -1)$, eine Translation um den Vektor $(-1, -2, -1)$.
- $R_z\left(\frac{\pi}{2}\right)$, eine Rotation um die z-Achse mit dem Winkel $\frac{\pi}{2}$.
- $M = S\left(1, \frac{1}{2}, 2\right) \cdot T(-1, -2, -1) \cdot R_z\left(\frac{\pi}{2}\right)$, die Gesamttransformationsmatrix

$S\left(1, \frac{1}{2}, 2\right) =$

$T(-1, -2, -1) =$

$R_z\left(\frac{\pi}{2}\right) =$

$M = S\left(1, \frac{1}{2}, 2\right) \cdot T(-1, -2, -1) \cdot R_z\left(\frac{\pi}{2}\right) =$

Aufgabe 4 (4 + 2 + 6 = 12 Punkte)

Zeichnen Sie die folgenden vier Vertices in ein **rechtshändiges** Koordinatensystem ein (x-Achse zeigt nach rechts, y-Achse nach oben) und verbinden Sie sie zu einem Tetraeder. **Hinweis:** Ein Tetraeder ist eine Pyramide mit einer dreieckigen Grundfläche und besteht somit aus exakt **4 Dreiecken**. Jeder Punkt ist mit jedem anderen verbunden.

- Position: (0,0,-1), Farbe: (1,0,0,1)
- Position: (1,0,-2), Farbe: (1,1,0,1)
- Position: (0,1,-2), Farbe: (0,0,1,1)
- Position: (0,0,-2), Farbe: (0,1,0,1)

Zeichnung:

Geben Sie den Inhalt eines geeigneten Floatbuffers und des zugehörigen Intbuffers an, die als Vertex- und Indexbuffer die Vertices zu einem Tetraeder verbinden.

Vertexbuffer
Vertexlayout: { pos.x, pos.y, pos.z, color.r, color.g, color.b, color.a }

Indexbuffer
Topologie: GL_TRIANGLE_STRIP (-2 Punkte, bei GL_TRIANGLES)

Aufgabe 5 (3 + 4 + 5 = 12 Punkte)

- Erweitern Sie Abbildung 1 um diejenigen Vektoren, die zur Berechnung des diffusen Anteils des Phong Beleuchtungsmodells benötigt werden. Markieren Sie außerdem den Winkel, der letztendlich für die Stärke der diffusen Beleuchtung verantwortlich ist.
- Erweitern Sie Abbildung 2 um diejenigen Vektoren, die zur Berechnung des spekularen Anteils des Phong Beleuchtungsmodells benötigt werden. Markieren Sie außerdem den Winkel, der letztendlich für die Stärke der spekularen Beleuchtung verantwortlich ist.
- Erweitern Sie die GLSL Methode `vec4 Enlight(vec3 position, vec3 normal)`, die sowohl den diffusen als auch den spekularen Beleuchtungsanteil berechnet und deren Summe zurückgibt. Die Entfernung und Art der Lichtquelle soll dabei unberücksichtigt bleiben.

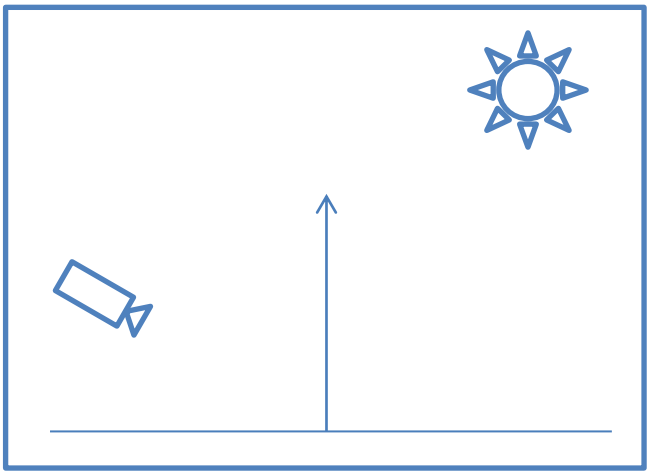


Abbildung 1

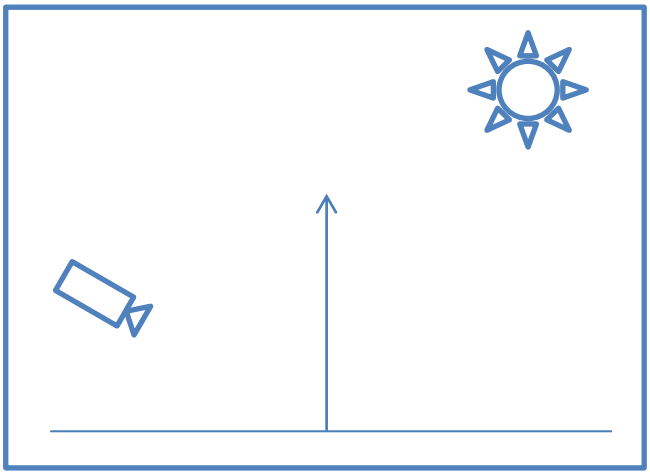


Abbildung 2

```
uniform vec3 lightPosition;
uniform vec3 eyePosition;
uniform vec3 I_d, I_s, c_d, c_s, k_c, k_s;
uniform float es;

vec4 Enlight(vec3 position, vec3 normal)
{









}
```

Aufgabe 6 (3 + 3 + 2 + 2 = 10)

Wir betrachten uns einen bestimmten Pixel auf dem Bildschirm. Folgender Ablauf von Ereignissen findet statt. Ergänzen Sie in den jeweiligen Kästen die aktuellen Pixeleigenschaften.

1. Aufruf: `glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA);`
2. Aufruf: `glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);` // `z` wird auf `1` und `color` auf `(0,1,1,1)` gesetzt.
3. Aufruf: `glEnable(GL_BLEND);`
4. Aufruf: `glEnable(GL_DEPTH_TEST);`
5. Drawaufruf erzeugt Fragment (`z = 0.6, color = (1, 1, 0, 0.5)`)

   Aktuelle Pixeleigenschaften:

   `z =` `color =`

6. Aufruf: `glDisable(GL_DEPTH_TEST);`
7. Drawaufruf erzeugt Fragment (`z = 0.7, color = (0, 1, 1, 0.5)`)

   Aktuelle Pixeleigenschaften:

   `z =` `color =`

8. Aufruf: `glEnable(GL_DEPTH_TEST);`
9. Aufruf: `glDisable(GL_BLEND);`
10. Drawaufruf erzeugt Fragment (`z = 0.1, color = (0.5, 0.5, 1, 0.2)`)

    Aktuelle Pixeleigenschaften:

    `z =` `color =`

11. Drawaufruf erzeugt Fragment (`z = 0.2, color = (0.75, 0.15, 0, 0.3)`)

    Aktuelle Pixeleigenschaften:

    `z =` `color =`

Aufgabe 7 (3 + 3 + 3 = 9 Punkte)

Wir sind uns in einem Programm nicht sicher, ob die Normalen einer Geometrie korrekt berechnet wurden. Um dies zu prüfen, würden wir gerne die Fragments entsprechend ihrer Normale einfärben. Implementieren Sie dazu den Vertex- und den Fragmentshader. **Ob8:** Die Einträge des Normalenvektors können natürlich auch negativ sein. Sorgen Sie für eine entsprechende Normierung.

Vertexshader:

```
uniform mat4 viewProjection, model, modelIT;

in vec3 positionMC;
in vec3 normalMC;



void main(void)
{




}
```

Fragmentshader:

```



void main(void)
{



}
```

Erklären Sie kurz, wie Sie die Farben des gerenderten Bildes zu interpretieren haben.

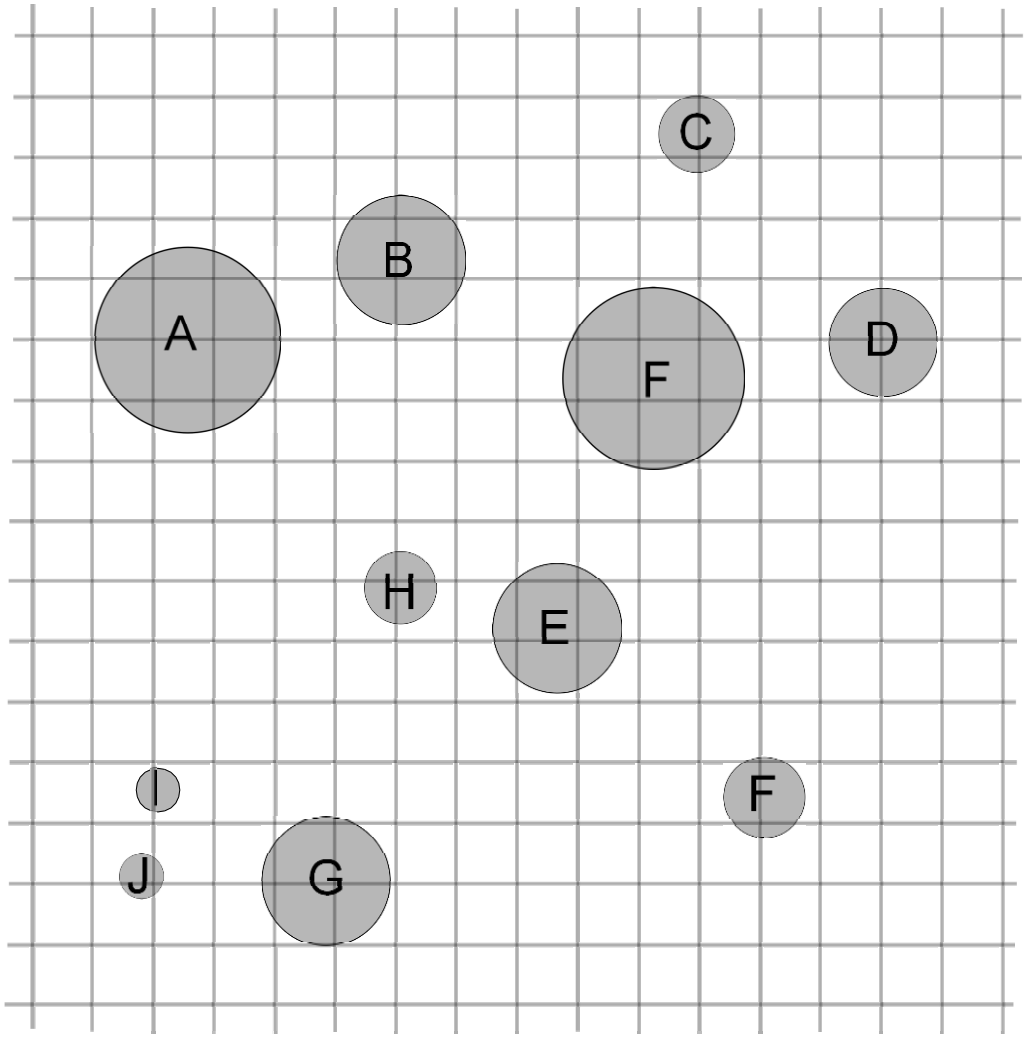
Erklärung:

Aufgabe 8 (5 Punkte)

Zeichnen Sie in das Diagramm die Struktur eines darauf aufbauenden zweidimensionalen KD-Tree ein. Dabei sollen folgende Regeln befolgt werden.

- Maximale Tiefe: 4
- Angestrebte Anzahl der Elemente pro Node: 2
- Teilung in jedem Schritt wechselnd zwischen Parallele zur x- bzw. y-Achse

Mit welche Achse beginnen Sie und warum?



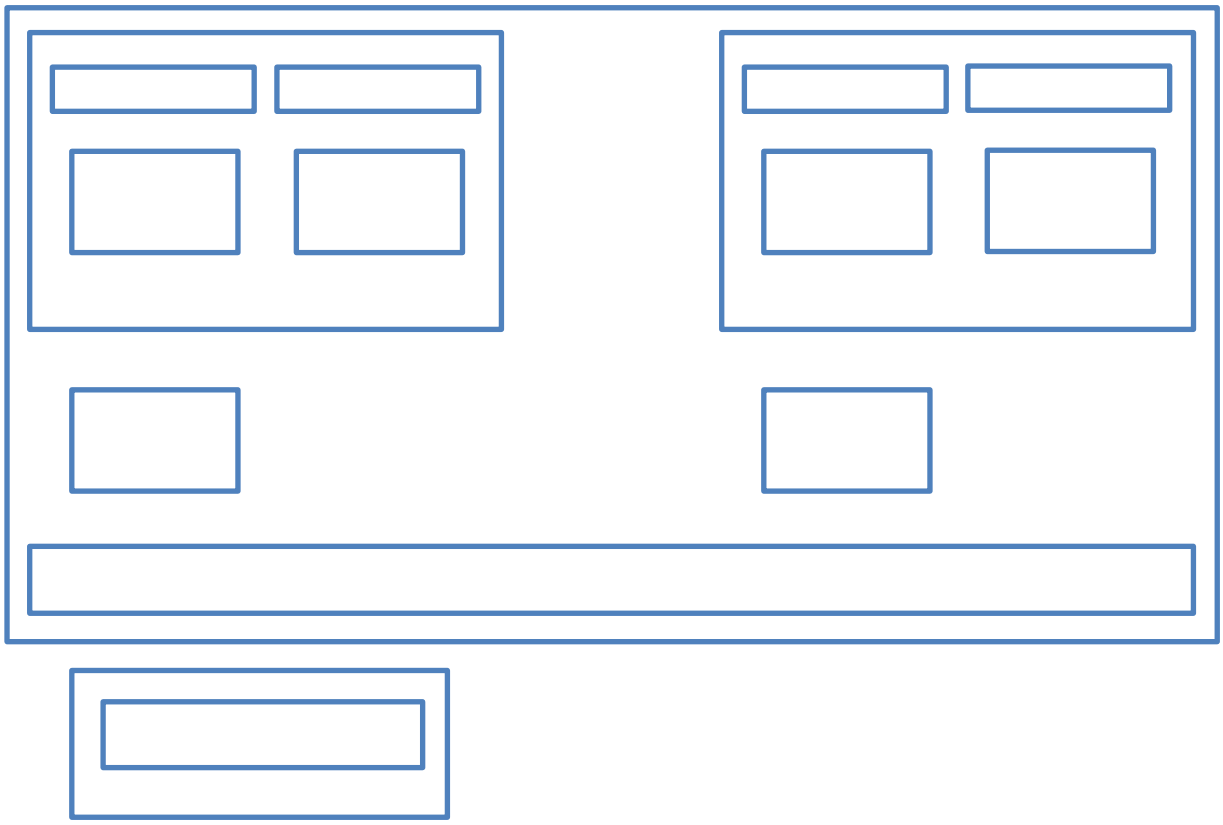
Aufgabe 9 (6 Punkte)

Warum muss der Parameter eines Sichtstrahls in einem KD-Raytracer nach vorne (t_{min}) und hinten (t_{max}) beschränkt sein? Begründen Sie dies mithilfe einer Zeichnung.

Aufgabe 10 (6 Punkte)

Vervollständigen Sie das unten stehende Diagramm dahingehend, dass es das Speichermodell von OpenCL widerspiegelt. Verwenden Sie Pfeile um das Lesen und Schreiben der verschiedenen Speicher zu symbolisieren und folgende Begriffe.

- Host
- Device
- Work Item
- Work Group
- Global Memory
- Local Memory
- Private Memory
- Host Memory



Aufgabe 11 (5 Punkte)

Erläutern Sie die Möglichkeiten, in OpenCL zu synchronisieren.

Aufgabe 12 (10 Punkte)

Erläutern Sie die Funktion des folgenden Kernels und begründen Sie den Synchronisationspunkt.

```
kernel void Riddle(global float* h, local float* j)
{
  uint id = get_global_id(0);
  j[get_local_id(0)] = h[id];
  barrier(CLK_LOCAL_MEM_FENCE);
  h[id] = j[get_local_size(0) - get_local_id(0) - 1];
}
```