Computer Grafik Blatt 6

June 2023

Aufgabe 1.

a)

	attribute	uniform	varying
Wert kann im Vertex Shader gelesen werden	✓	✓	✓
Wert kann Vertex Shader geschrieben und geändert werden			✓
Kann im Vertex Shader innerhalb einer Funktion definiert werden			
Hat im Vertex Shader grundsätzlich für jeden Vertex den gleichen Wert		✓	
Wert kann im Fragment Shader gelesen werden		✓	✓
Wert kann im Fragment Shader geschrieben und geändert werden			
Kann im Fragment Shader innerhalb einer Funktion definiert werden			
Hat im Fragment Shader grundsätzlich für jedes Fragment des selben Dreiecks den gleichen Wert		✓	
Wert kann im Hauptprogramm geschrieben und geändert werden	✓	✓	

b)

Die finale Position eines Vertex muss im Vertex Shader in die eingebaute Variable gl_Position geschrieben werden.
Die finale Farbe eines Fragments muss im Fragment Shader in die eingebaute Variable gl_FragColor geschrieben werden.

c)

- ☐ ModelView-Transformation

☐ Projektion

☑ Rasterisierung

☐ z-Buffer-Test

d)

$4096 \cdot 2304 = 9437184$ Pixel. Der Fragment Shader wird somit mindestens 9437184 mal aufgerufen. Gibt es mehrere überlappende Objekte, so wird der Fragment Shader öfter aufgerufen. (??)

e)

1. vec4 v = vec4(1.0, 2.0, 3.0, 4.0);
2. float x = 8;
   - 8 ist ein integer, 8 als float wäre 8.0
3. vec3 v1 = v.xxx;
4. vec3 v2 = v.xyzw;
   - vec3 bekommt vec4 zugewiesen
5. float f = v[1];
6. v[0] = f;
7. vec3 w = vec3(5.0, 6.0, 7.0);
8. mat2 m = mat2(2.0, 3.0, 1.0, 0.0, 0.1);
   - mat2 bekommt 5 Werte übergeben, obwohl nur 4 benötigt werden
9. vec3 v3 = v.gb;
   - vec3 wird vec2 zugewiesen
10. vec3 v4 = v.abc;
    - abc sind keine gültigen Werte
11. vec3 d = dot(w, v1);
    - dot() gibt float zurück und nicht vec3
12. vec3 e = dot(v.ab, w.xy);
    - dot() gibt float zurück und nicht vec3
13. vec3 r = cross(w, v);
    - cross() von vec3 und vec4 nicht möglich
14. v[1] = x;

f)

Der Fragment Shader wird für jedes Pixel ausgeführt, aber nur die Pixel, welche im Dreieck liegen, werden gezeichnet. Das Dreieck nimmt bei den gegebenen Koordinaten ca. 1/4 des Bildschirms ein, also werden ca. 25% der Pixel nicht mehr schwarz sein. (??)

Das Dreieck wird bei der Rasterisierung geclipt. Vermutlich auf (0,0,0,1),(1,-1,0.25,1),(-1,-1,0.0625,1). Danach wird für jedes Fragment des Dreiecks die Farbe auf (0,0,1,1) also Blau mit voller Opazität gesetzt. Nach dem Fragmentshader wird dann für jeden Pixel der Verdeckungstest gemacht, und da das Dreieck vollständig vor dem maximalwert 1 liegt werden alle pixel die fragmente des Dreieck enthalten Blau bemalt. Alle übrigen Pixel werden Schwarz bemalt, da hier die im Color-Buffer gesetzte Frabe genutzt wird. Final erhalten wir ein blaues Dreieck von der mitte zu beiden unteren Ecken mit einem schwarzen Hintergrund. Somit sind ziemlich genau 25% der Pixel nicht mehr Schwarz.