Karlsruher Institut für Technologie
Lehrstuhl für Computergrafik

Prof. Dr.-Ing. Carsten Dachsbacher
Dipl.-Inf. Christoph Schied

3. Übungsblatt zur Vorlesung Interaktive Computergrafik im SS 2017

Besprechung am **Mittwoch, 14.06.2017**, 10:30 Uhr.

Aufgabe 1 *Profiling*

Die Grafikkarte läuft asynchron zur CPU, wobei eine Work-Queue zur Kommunikation benutzt wird. Soll die Performance verschiedener Algorithmen und Implementierungen evaluiert werden muss man entscheiden welche Größen gemessen werden sollen. Hierfür können Timer sowohl auf CPU und GPU Seite benutzt werden. Werden CPU-seitige Timer benutzt, so wird—durch die asynchrone Kommunikation—gemessen in welcher Zeit die Draw-Calls abgesetzt werden. Durch GPU-seitige Timer wird hingegen die Abarbeitung der Work-Queue durch die GPU gemessen.

Zur Zeitmessung auf der CPU können Sie std::chrono (`http://en.cppreference.com/w/cpp/chrono`) verwenden. GPU-Timer können in OpenGL mittels *Timer Queries* (siehe `https://www.khronos.org/opengl/wiki/Query_Object` und `https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_timer_query.txt`) abgefragt werden. Mittels `gl_Finish` kann die CPU darauf warten dass die GPU die gesamte Work-Queue abarbeitet.

In dieser Aufgabe soll eine bestehende Implementierung eines separierten Blurs evaluiert werden, wobei die Gesamtzeit sowie die Zeiten für die einzelnen Schritte gemessen werden sollen. Annotieren Sie die Implementierung des separierten Blurs in `Blur::blur()` mit CPU und GPU Timern und vergleichen Sie die Resultate! Wie wirkt sich die Synchronisierung der Pipeline (Synchronisierung jeweils vor dem auslesen der CPU Timer) auf die Messungen aus?

Zur Ausgabe der Timer in der GUI können Sie die Funktion

```
ImGui::Text(const char* fmt, ...);
```

benutzen, welche `printf`-style Format-Strings aktzeptiert.

Aufgabe 2 *Optimierung*

Erstellen Sie eine optimierte Implementierung und vergleichen Sie mit der initialen Implementierung. Lässt sich die Performance dadurch verbessern, den Filterradius fest im Shader vorzugeben? Wie lässt sich ein eventueller Gewinn erklären? Wie wirkt sich das Format des Framebuffers auf die Performance aus? (Vergleichen Sie GL_RGBA32F, GL_RGBA16F, GL_RGBA8). Können Teile der Berechnung auf den Vertex Shader, oder in Uniform Variablen verlagert werden?

Benutzen Sie die Technik, die Sie in der Ubung kennengelernt haben, um die Anzahl der Texturzugriffe durch Ausnutzung der (bi-)linearen Texturfilterung zu reduzieren! Eine Beschreibung für einen derartig optimierten Blur finden Sie unter `http://rastergrid.com/blog/2010/09/efficient-gaussian-blur-with-linear-sampling/`.

Framework

Wir stellen für jedes Übungsblatt ein Framework bereit. Das Framework nutzt C++ 11 und wird unter Linux getestet. Es ist allerdings auch unter Windows mit Visual Studio 2013 lauffähig. Die Datei `Kompilieren.txt` enthält Informationen darüber, wie sie das Framework kompilieren.

Hinweis: Wenn sie Visual Studio verwenden sollten Sie das Projekt im Release oder Release-WithDebug starten um die Ladezeiten zu reduzieren.