Framebuffer Objects

Framebuffer Objects (FBOs) erlauben es, benutzerdefinierte Framebuffer zu erstellen, um den Output eines Render Vorgangs aufzufangen – für uns hauptsächlich deshalb nützlich, weil wir FBOs benutzen können, um das Ergebnis eines Render Vorgangs in eine Textur umzuleiten.

Unterschiede zwischen einem Framebuffer Object und dem Default-Framebuffer:

Der RGBA Buffer des Default-Framebuffers stellt grundsätzlich 8 Bit pro Farbkanal zur Verfügung, d.h. Farb- und Alphawerte werden im Wertebereich [0, 255] gespeichert. (Wenn Sie im Shader etwas wie: `fragmentColor = vec4(...);` schreiben, werden die float Werte entsprechend konvertiert). FBOs sind viel flexibler, und ermöglichen es z.B., floats zu speichern – was dann natürlich mit entsprechendem Speicherverbrauch verbunden ist.

Beim Rendern in den Default Framebuffer schreiben wir im Fragment Shader typischerweise nur einen Wert. FBOs erlauben es uns, mehrere Werte pro Fragment in verschiedene Texturen zu speichern (Stichwort Multiple Render Targets, kurz MRT).

Während der Default Framebuffer eine eingebaute Multisampling-Kapazität hat, sind die Texturen, in die wir bei der Verwendung von FBOs rendern, zunächst einmal nicht multisampling-fähig; Tatsächlich ist Anti-Aliasing in Verbindung mit FBO-Techniken ein wichtiges Forschungsgebiet.

Schritte zum Erzeugen & Initialisieren eines FBOS:

Schritt 1: 0...N leere 2D Texturen erstellen. (Die maximale Anzahl von Texturen, welche an ein FBO attached werden können, ist von der Grafikkarte abhängig, es sind aber zumindest 8) Falls der Z-Test beim Rendern benötigt wird, eine eigene 2D Textur mit Tiefenformat erstellen.

Schritt 2: Das Framebuffer Object erstellen und als GL_DRAW_FRAMEBUFFER binden.

Schritt 3: Die Texturen an das Framebuffer Object attachen. RGBA Texturen an GL_COLOR_ATTACHMENT0...GL_COLOR_ATTACHMENTN, die Tiefentextur an GL_DEPTH_ATTACHMENT.

Schritt 4: Die Draw Buffers des Framebuffer Objects richtig konfigurieren. Die Draw Buffers regeln das Zusammenspiel zwischen den Framebuffer Attachments und den Outputs das Fragment Shaders, also dass die im Fragment Shader berechneten Werte in die richtigen Texturen geleitet werden.

Schritt 5: Zur Sicherheit einen Completeness Check durchführen. Falls dieser fehlschlägt, nachlesen, was der Error Code bedeutet. (https://www.opengl.org/wiki/Framebuffer_Object#Framebuffer_Completeness)

Verwendung eines FBOs (Rendern):

Schritt 1: FBO binden.

Schritt 2: Szene rendern – wie immer.

Schritt 3: Default Framebuffer binden – hat immer das Handle 0. Oder evtl. anderes FBO binden.

Nach dem Rendern können Sie mit den in der Textur gespeicherten Daten verfahren, wie Sie möchten. Der häufigste Fall ist sicher, ein Postprocessing auf die Texturen anzuwenden – auch das Deferred Shading, kann in gewisser Weise als Postprocessing aufgefasst werden. **In so einem Fall wird wie folgt vorgegangen:**

Schritt 1: (optional)Tiefentest ausschalten

Schritt 2: FBO Textur(en) binden & an Shader übergeben

Schritt 3: Ein Fullscreen Quad, d.h. Ein Quad mit Vertices { (-1, -1, 0), (1, -1, 0), (1, 1, 0), (-1, 1, 0) } rendern. Keine Model-, View- oder Projection Matrix anwenden! Dies bewirkt, dass das Quad den gesamten Viewport exakt ausfüllt.

Schritt 4: Im Fragment Shader die Texturen auslesen... entweder mit `texelFetch` und gl_FragCoord, oder mit `texture` und UV Koordinaten.

Sie finden insgesamt 3 Samples zur Verwendung von FBOs:

Eine Farbtextur + eine Tiefentextur: Szene wird in ein FBO gerendert; danach in einem Postprocessing Schritt gefiltert. Aus diesem Sample ersehen Sie:

- Wie die Initialisierung eines FBOs abläuft.
- Rendern mit zwei Renderpasses, mit unterschiedlichen Shadern / States.
- Wie man im zweiten Renderpass die Farbtextur des FBOs verwendet.
- Wie man ein Fullscreen Quad möglichst einfach rendern kann. (C++ Code & Vertex Shader des zweiten Renderpasses)
- Den Unterschied zwischen Texturzugriffen mit texture und texelFetch (Fragment Shader des 2ten Renderpasses)

Nur Tiefentextur: Im Sample werden einfach die Tiefenwerte ausgegeben, und gegebenenfall linearisiert. Aus dem Sample ersehen sie:

- Wie man die Draw Buffers für so ein FBO konfiguriert.

- Wie der Fragment Shader aussieht, wenn man keine Farbwerte berechnet.

Mehrere Farbtexturen + eine Tiefentextur. Aus dem Sample ersehen Sie:

- Das Zusammenspiel zwischen den Draw Buffers des FBOs und den Fragment Shader Outputs.
- Wie man im zweiten Renderpass mehrere Texturen gleichzeitig bindet und an den Shader übergibt.