Motion blur

Wstęp

Motion blur to zbiór technik których celem jest odtworzenie pozornego rozmazania występującego podczas obserwowania szybko poruszających się obiektów. Od dawna twórcy gier komputerowych starają się odtworzyć ten efekt na różne sposoby – a gracze na ogół wyłączali go gdy tylko było to możliwe. Dzisiaj skupimy się na technice postprocessingowej, którą można wykorzystać zarówno w pipelinach tradycyjnych jak i wykorzystujących deferred shading.

Opis metody

Podczas renderowania danej klatki stworzymy dodatkowo bufor zawierający informacje o prędkości wszystkich obiektów względem kamery. Następnie, po wyrenderowaniu klatki z oświetleniem, wykorzystamy bufor prędkości aby rozmazać klatkę zgodnie z kierunkiem i wartością prędkości w każdym miejscu.

Etap 1: Dodanie nowego etapu renderowania

W stanie obecnym są dwa etapy renderowania. Pierwszy zbiera dane o geometrii w teksturach (w dwóch programach – pierwszy rysuje poruszający się obiekt, a następny otaczający go skybox), drugi cieniuje obraz przed umieszczeniem na ekranie. Ponieważ chcemy dodać kolejny etap, dane z etapu oświetlenia będzie trzeba zapisać w teksturze. Oznacza to wygenerowanie nowego bufora, który będzie przechowywał wynik drugiego etapu renderowania.

Można to zrobić w miejscu TODO1, w oparciu o stworzony wyżej bufor na potrzeby pierwszego etapu. Potrzebna będzie zaledwie jedna tekstura na kolor, bez bufora głębokości.

Stworzenie trzeciego etapu renderowania wymaga oddzielnego zestawu shaderów. Można wykorzystać istniejący vertex shader, bo postprocessing opiera się na rysowaniu jednego quada. Natomiast fragment shader wymaga skopiowania i zmodyfikowania odpowiedniego shadera z drugiego etapu.

W miejscu TODO2 trzeba stworzyć nowy program wykorzystujący te dwa shadery.

Aby zobaczyć efekty (czyli że nic nie zepsuliśmy), trzeba jeszcze zmienić sam proces renderowania. Na początku drugiego etapu renderowania trzeba użyć stworzonego przez nas bufora. Poniżej trzeba dodać nowy etap, którego efekty zostaną wyświetlone na ekranie (bufor trzeba ustawić na 0). Będzie on o wiele prostszy od poprzednich etapów, ponieważ wykorzystuje tylko jedną teksturę – stworzoną w etapie drugim teksturę kolorów.

Etap 2: Wygenerowanie bufora prędkości

Aby rozmazać obraz potrzebujemy najpierw informacji o tym jak wszystko się porusza. Wygenerujemy te dane podczas pierwszego etapu renderowania i zapiszemy je w teksturze. Wymaga to stworzenia nowej tekstury i dołączenia jej do buforu w miejscu TODO3 (oraz drobnej zmiany w miejscu TODO4). Ta tekstura potrzebuje tylko dwóch kanałów.

Wyliczenie prędkości obiektów na ekranie względem kamery jest najbardziej skomplikowanym etapem tego podejścia do motion blur. Trzeba w tym celu stworzyć macierze przechowujące pozycję

poruszającego się obiektu oraz poprzednią pozycję kamery w postaci macierzy. Zmienne te trzeba stworzyć tak, aby przetrwały pomiędzy pętlami.

Macierz obiektu można zapisać w miejscu TODO5, a macierz kamery w miejscu TODO6.
Obie te macierze trzeba przekazać do programów odpowiedzialnych za model oraz skybox. Jest to do zrobienia w części odpowiedzialnej za pierwszy etap renderowania.

Następnie należy zmodyfikować shader wierzchołków pierwszego etapu aby obliczył on pozycję w przestrzeni ekranu zarówno w obecnej jak i poprzedniej klatce. Pozycje te trzeba przekazać do fragment shadera, znormalizować i zapisać ich różnicę w teksturze prędkości. W tym momencie wartościowym eksperymentem może być wypisanie wartości prędkości do tekstury koloru. Będzie to wymagało kreatywnego przeskalowania aby uzyskać widoczne wyniki.

Etap 3: Rozmazanie.

Wielokrotne próbkowanie bufora koloru.

Teksturę prędkości trzeba wykorzystać teraz do odpowiedniego rozmazania obrazu w trzecim etapie. Wymaga to najpierw podłączenia tekstury prędkości do trzeciego programu. Następnie, zamiast pobierać z tekstury koloru jedną wartość, należy pobrać ich kilka, za każdym razem przesuwając się zgodnie z prędkością w punkcie początkowym. Próbki należy dodać z odpowiednimi wagami, które sumują się do jedynki. Przykładowymi wagami dla czterech próbek są 0.4, 0.3, 0.2 i 0.1. Prędkość należy przeskalować do uzyskania zadowalającego efektu. W momencie gdy próbki pobierać będziemy spoza tekstury koloru, mogą pojawić się niepożądane artefakty. Nie istnieje proste i idealne rozwiązanie tego problemu (poza potencjalnie renderowaniem całej klatki w większej rozdzielczości, tworząc margines), ale warto pobawić się opcjami tej tekstury, aby osiągnąć najmniej rażące artefakty (warto poszukać informacji o funkcji glTexParameter).