

Projekt: rendering obrazów - raport

Michał Jaworski

21 stycznia 2018

W tym raporcie opisałem istotniejsze problemy, jakie napotkałem podczas pisania projektu, wraz z użytymi przeze mnie rozwiązaniami.

1 Reprezentacja sceny

Pierwszym istotnym problemem było zaprojektowanie odpowiedniego formatu tekstowego do reprezentacji sceny. Format ten powinien być przede wszystkim czytelny i łatwy w użyciu. Musi on także zapewniać możliwość łatwego rozszerzania, w przypadku dodania do programu nowych obiektów, rodzajów powierzchni itp.

Opracowany przeze mnie format został szczegółowo omówiony w dokumentacji projektu. W celu ułatwienia procesu parsowania użyłem biblioteki Parsec, która dostarcza wielu funkcji pomocnych podczas tworzenia parserów.

2 Typy danych

Istotnym problemem podczas pracy nad projektem było stworzenie odpowiednich typów danych oraz klas typów. W moim programie zdecydowałem się na reprezentację większości typów związanych z opisem sceny jako abstrakcyjnych typów danych. Umożliwia to reprezentację zawartości sceny w postaci list. Z kolei kolory stanowią w moim programie klasę typów. Dzięki temu można łatwo dodać kolejne sposoby ich reprezentacji oraz wyrenderować scenę z użyciem dowolnego z nich.

3 Umiejscowienie kamery

Kolejnym problemem, jaki napotkałem, było odpowiednie ustawienie kamery. Aby maksymalnie uprościć obliczenia, jak również zapewnić użytkownikowi łatwe rozmieszczanie obiektów na scenie zdecydowałem się rozwiązać ten problem w sposób następujący:

- Kamera patrzy w kierunku wyznaczonym przez wektor $(0, 0, 1)$.
- Środek prostokąta widoku znajduje się w punkcie $(0, 0, 0)$.
- Ognisko widoku znajduje się w punkcie $(0, 0, -depth)$.

4 Wyznaczanie cieni

Aby sprawdzić, czy dany punkt rzeczywiście otrzymuje światło z badanego źródła, program wyznacza dodatkowy promień, o początku w badanym punkcie i kierunku przeciwnym do kierunku padania promieni. Badając punkty przecięcia tego promienia można ustalić, czy ten punkt znajduje się w cieniu innego obiektu. Niestety, ze względu na niedokładności arytmetyki maszynowej promień ten przecinał się niekiedy z badanym obiektem, tworząc niepoprawne cienie. Problem ten rozwiązało przesunięcie początku promienia o niewielki ułamek wektora normalnego badanej powierzchni.

5 Odbijanie promieni

Implementując powierzchnie odbijające światło, uczyniłem śledzenie promieni procesem rekurencyjnym. Dzięki temu, aby wyznaczyć kolor danego punktu powierzchni lustrzanej wystarczy rekurencyjnie wyznaczyć kolor obiektu, którego obraz odbija się w badanym punkcie, rekurencyjnie śledząc odpowiednio wyznaczony promień odbity. Niestety, takie podejście dopuszcza możliwość nieskończonej rekursji (np. gdy scena składa się z dwóch równoległych, lustrzanych płaszczyzn). Aby temu zapobiec ograniczyłem głębokość rekurencyjnych wywołań funkcji. Domyślnie dopuszczalne są maksymalnie 4 wywołania, użytkownik może jednak ustawić własną wartość w nagłówku pliku opisującego scenę. W przypadku przekroczenia maksymalnej głębokości rekursji zwracany jest kolor tła (tzn. promień uznaje się za niekolidujący z żadnym obiektem sceny).