Projekt: rendering obrazów - raport

Michał Jaworski

9 stycznia 2018

1 Reprezentacja sceny

Istotnym problemem, jaki napotkałem podczas pisania programu, było zaprojektowanie odpowiedniego formatu tekstowego do reprezentacji sceny. Format ten powinien być przede wszystkim czytelny. Musi on także zapewniać możliwość łatwego rozszerzania, w przypadku dodania do programu nowych obiektów, rodzajów powierzchni itp.

Zaprojektowany przeze mnie format wygląda następująco:

- Wiersze rozpoczynające się od znaku # są traktowane jako komentarze i ignorowane przez parser.
- Wielkość liter w słowach kluczowych nie ma znaczenia
- Wektory oraz punkty w przestrzeni trójwymiarowej przedstawione są jako trzy liczby rzeczywiste oddzielone białymi znakami
- W podobny sposób reprezentowane są kolory w postaci RGB, kolejne liczby oznaczają wartości na odpowiednich kanałach
- W parserze zdefiniowano również stałe reprezentujące podstawowe kolory. Są to: *black*, *white*, *red*, *green*, *blue*, *cyan*, *magenta* oraz *yellow*

Plik tekstowy opisujący scenę składa się z następujących części:

1. **Nagłówek** (obowiązkowy), każda z poniższych informacji poprzedzona jest odpowiednim słowem kluczowym:
   - *imWidth* oraz *imWeight*: liczby całkowite reprezentujące odpowiednio szerokość oraz wysokość obrazka w pikselach, podanych jako liczby całkowite
   - *canvWidth* oraz *canvHeight*: liczby rzeczywiste reprezentujące odpowiednio szerokość oraz wysokość prostokąta, przez który obserwowana jest scena. podanych jako liczby całkowite
   - *depth*: liczba rzeczywista reprezentująca odległość ogniska od prostokąta
   - *bgcolor*: opcjonalny parametr oznaczający kolor tła. Domyślnie jest to kolor czarny

- *raydepth*: opcjonalny parametr określający maksymalną głębokość rekursji podczas śledzenia promieni. Domyślną wartością jest 4

2. **Źródła światła** (opcjonalne). Lista źródeł światła rozpoczynająca się słowem kluczowym *lights*. Dostępne są następujące źródła światła:

   - *directional i c d*: kierunkowe źródło światła o intensywności wyznaczonej przez liczbę rzeczywistą i, o kolorze c, świecące w kierunku wskazywanym przez wektor d
   - *spherical i c x*: punktowe źródło światła o intensywności i, kolorze c, znajdujące się w punkcie x

3. **Obiekty** (opcjonalne). Lista znajdujących się na scenie obiektów. Każdy obiekt przedstawiony jest w postaci *kształt powierzchnia*. Program udostępnia następujące kształty:

   - *sphere x r*: kula o środku x i promieniu r
   - *plane x d*: płaszczyzna zawierająca punkt x o wektorze normalnym równoległym do wektora d

   oraz następujące rodzaje powierzchni:

   - *diffusive c*: powierzchnia rozpraszająca światło, w kolorze c
   - *reflective*: powierzchnia odbijająca światło
   - *luminous c*: powierzchnia świecąca własnym światłem, w kolorze c
   - *mixed* t_1 s_1 $[t_2]$ $[s_2]$ ...: powierzchnia mieszana, gdzie wartości t_i wyznaczają proporcje, a s_i to mieszane rodzaje powierzchni

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. Umożliwia to łatwe dodanie kolejnych sposobów ich reprezentacji oraz wyrenderowanie sceny z dowolnie wybranym 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 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).