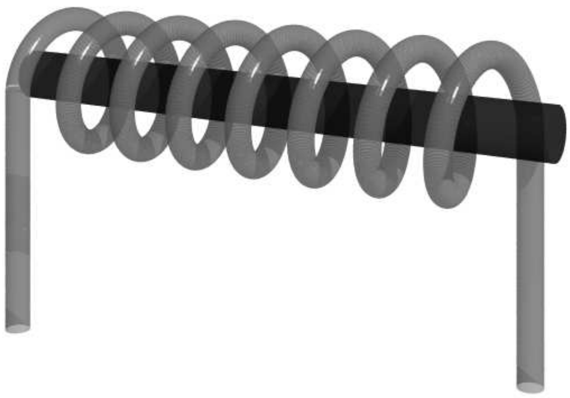
POV-Ray. Co to jest?

Piotr Strzelczyk

Notatka ta powstała jako drukowana wersja mojego referaciku wygłoszonego na ostatniej Konferencji w Bachotku.

Istnieje wiele metod tworzenia grafiki trójwymiarowej. Od najprostszych grafik wektorowych typu szkieletowego, poprzez obrazy składające się z wypełnionych jednolitym kolorem ścian, aż po bardziej zaawansowanie techniki cieniowania wielościanów. Artykuł ten dotyczy techniki tworzenia obrazów, które stwarzają czasem wrażenie bardziej realnych niż fotografie. Metoda ta nazywa się śledzeniem promieni (*ray-tracing*). Rysunki stworzone tą techniką są łatwe do zauważenia i bardzo efektowne. Można poznać je po charakterystycznym uproszczeniu – geometryzacji obrazu rzeczywistości przy jednoczesnym perfekcyjnym oddaniu wykończenia powierzchni, jej cieniowania, odblasków, odbić, rozproszeń i innych efektów świetlnych. Powstają także bardzo ładne animacje tworzone za pomocą programów do ray-tracingu.



Promień światła tworzy obraz

Pora chyba już wyjaśnić na czym polega metoda śledzenia promieni. Jak większość dobrych koncepcji jest ona niezwykle prosta. Opiera się na symulacji za pomocą komputera optycznego modelu naszego otoczenia. Po prostu obliczamy dla najmniejszego promyczka światła całą jego drogę od źródła do oka obserwatora. Oczywiście trzeba przy tym uwzględnić jak najwięcej praw optyki, zwłaszcza rozpraszanie światła na różnych przedmiotach, oraz zmiany jego barwy w zależności od tego, co napotkało na swej drodze.

Jak można sobie wyobrazić jest to potwornie złożone zadanie obliczeniowe, i dlatego właśnie dopiero od niedawna można używać tej metody także w warunkach domowych. Obliczenia dotyczące całego obrazka zajmują do kilku godzin pracy komputera klasy Pentium. Ponieważ kalkulacje dotyczące odbicia, rozproszenia, zmiany barwy lub załamania promienia są zawsze wykonywane na liczbach zmiennoprzecinkowych, więc koprocesor arytmetyczny jest praktycznie niezbędny.

Śledzenie promieni wymagałoby jeszcze większych mocy obliczeniowych, gdyby nie wymyślono chytrego sposobu, który umożliwia obliczenie całego obrazu w skończonym czasie.

Każde źródło światła wysyła bowiem nieskończenie wiele promieni w różnych kierunkach. Nawet dwa bardzo bliskie siebie promienie mogą przebiegać diametralnie różną drogę i trafiać w zupełnie inne miejsca. Śledzenie tak wielu promieni zajęłoby dużo więcej czasu.

Trick umożliwiający stosowanie metody ray-tracingu polega na śledzeniu promienia od końca, czyli od oka obserwatora. W takim przypadku wystarczy policzyć drogę tylko tylu promieni wpadających do oka obserwatora, z ilu punktów ma składać się nasz obraz. Wynika to z prostego faktu, że jeśli byśmy nawet obliczyli więcej promieni, to i tak nie byłoby jak ich przedstawić na obrazie.

Wyobraźmy sobie więc, że obraz tworzymy na szklanym prostokącie zawieszonym w pewnej odległości od obserwatora. Wystarczy teraz znaleźć drogę promienia przechodzącego przez kolejny kwadracik (*pixel*) na szybie i wpadającego do oka obserwatora, obliczyć z którego (lub których) źródeł światła został wysłany, od jakich przedmiotów się odbił, na jakich się rozproszył, aby określić jak będzie wyglądał ten punkt obrazu. Następnie należy powtórzyć ten proces dla wszystkich punktów ekranu, i już mamy kompletny obraz. Skończona liczba punktów obrazu wynika z pożądanej rozdzielczości grafiki i określa wprost czas trwania procesu obliczeniowego.

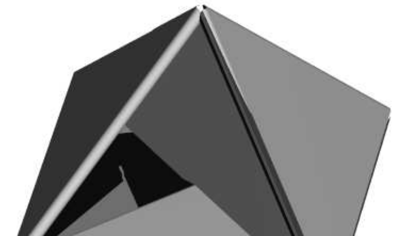
Z wstecznego śledzenia promieni wynika jedno ciekawe zjawisko. Mianowicie o ile w „świecie promieni” obserwator patrząc przez lupę widzi obraz przez nią powiększony, to niestety światło padające przez tę samą soczewkę ze źródła światła na przedmiot nie tworzy na nim nigdy plamki skupionego światła. Jest to jedno z ograniczeń realności generowanych obrazów, którymi płacimy za znaczne uproszczenie zadania obliczeniowego.

Co to właściwie jest POV-Ray?

Ponieważ mam niewiele miejsca, chciałbym więc teraz przejść trochę do praktyki. Programów do tworzenia obrazów metodą śledzenia promienia jest wiele, na przykład Vivid, PolyRay, POV oraz wiele innych. Tutaj opiszę bliżej tylko jeden z nich, mianowicie POV-Ray. Nazwa tego programu jest skrótem, i jak to najczęściej bywa jest to skrót nazwy angielskiej. Pełna nazwa brzmi: *Persistance of Vision Ray Tracer*, wyrażenie to trudno przetłumaczyć na język polski, ale znaczy mniej więcej „trwała wizja”. Autorem programu jest spory zespół (w dokumentacji wymienionych jest 17 autorów), głównymi autorami są Chris Young i Drew Wells. Artykuł ten napisałem opierając się na wersji 2.0 programu, choć widziałem już wersję 3.0, która zawiera wiele nowych możliwości (rozproszenia światła w powietrzu, mgła, płomienie oraz mnóstwo innych efektów, a także znacznie ulepszone łącze z użytkownikiem) i o której też chciałbym kiedyś napisać artykuł.

Dlaczego piszę właśnie o POV-ie? Po pierwsze jest to jeden z najlepszych i najbardziej uniwersalnych programów do ray-tracingu. Po drugie jest to program *public-domain*. Te dwa fakty powodują, że jest on chyba najpopularniejszy, bardzo dobrze przetestowany, i jak widać po pojawieniu się wersji 3.0, ciągle rozwijany. Wersje POV-Ray istnieją na większość platform sprzętowych od Amigi przez Apple i PC aż po maszyny UNIX-owe.

Do zalet POV-a należy zaliczyć fakt, że potrafi on w czasie tworzenia obrazu wyświetlać go na ekranie komputera, oraz że potrafi kontynuować przerwane obliczenia od miejsca gdzie je zakończył. Jeśli dodać jeszcze, że pliki wejściowe opisujące tworzone obrazy są czystymi plikami ASCII, a jednocześnie pewnym rodzajem programu, to już widać dlaczego wybrałem właśnie POV. Jednocześnie też można dostrzec pewne cechy wspólne z TEX-em, a uwzględniając możliwość umieszczania grafik stworzonych POV-em w dokumentach TEX-owych, informacja o tym programie staje się uzasadnionym elementem biuletynu GUST-u.



Powyższy rysunek został zainspirowany przez Keesa van der Laana, który przysłał nam papierowy model takiej bryły.

POV jako język programowania

POV pracuje, tak jak TEX, w trybie wsadowym. Na wejściu dostaje plik tekstowy, który opisuje świat jaki chcemy ujrzeć, a na wyjściu, po najczęściej długotrwałych obliczeniach, otrzymujemy plik graficzny zawierający wygenerowany obraz. Plik wejściowy POV-a składa się z ciągu instrukcji, które opisują obiekty zawarte w tworzonym świecie, jest to więc pewien rodzaj języka programowania. Język POV-a jest językiem notacyjnie troszkę podobnym do języka C, ale z powodu znacznej różnicy zadań do których są one przeznaczone jest to podobieństwo bardzo powierzchowne. Charakterystyczną cechą POV-a jest to, że jego język jest w gruncie rzeczy językiem niskiego poziomu, wręcz asemblerowym. Nie występują w nim żadne funkcje, pętle, konstrukcje warunkowe, a nawet praktycznie nie ma w nim zmiennych.

Program w POV-ie składa się z bloków, z których każdy opisuje jakiś obiekt z tworzonego świata. Takim obiektem może być jakaś bryła umiejscowiona w kreowanej przestrzeni, ale także źródło światła oświetlającego nasz nowy byt, lub punkt przestrzeni, z którego zobaczymy obraz. Następną cechą POV-a jest fakt, że w tym języku nie ma komend opisujących takie obiekty jak punkt czy prosta, można je przedstawić tylko jako małą kulkę lub sześcianik, czy długi, cienki walec. Jest to więc język opisu obiektów fizycznych, a nie matematycznych.

Króciutki wstęp do programowania POV-Ray

POV pozwala na dużą dowolność jeśli chodzi o formatowanie pliku wejściowego. Może on w ogóle nie zawierać odstępów, ale także zawierać dowolnie dużo spacji, wcięć, tabulatorów i nowych linii. Końce linii traktowane są jak spacje, a dowolna ilość spacji jak jedna spacja. Całkiem jak w TEX-u.

W programie POV-a można oczywiście umieszczać także komentarze. POV jest językiem prekompilowanym, czyli ilość komentarzy, sposób formatowania tekstu i długości identyfikatorów nie mają wpływu na czas generowania obrazu. Poprawiają one tylko czytelność programu.

Nie mogę teraz przeprowadzić pełnego kursu języka POV-a, ale ogólnie rzecz biorąc słowa kluczowe tego języka opisują kształt i istotę przedmiotu, a także jego kolor, wykończenie powierzchni, transformacje którym podlega itp.

A teraz trochę praktyki

Teraz spróbujmy stworzyć własny obrazek. Żeby uzyskać jakąkolwiek grafikę musimy opisać co najmniej trzy obiekty. Po pierwsze obiekt, który ma się ukazać na ekranie np.:

```
box {<0,0,0>, <1,1,1> pigment {Red}}
```

Jest to sześcian o przeciwległych wierzchołkach w podanych punktach, czyli o boku jednostkowym, z jednym wierzchołkiem w początku układu współrzędnych i o kolorze zadeklarowanym (np. czerwonym).

Po drugie, musimy zdefiniować jakieś źródło światła, aby nasz obraz nie tonął w ciemności:

```
light_source {<3,10,-20> color rgb<1,1,1>}
```

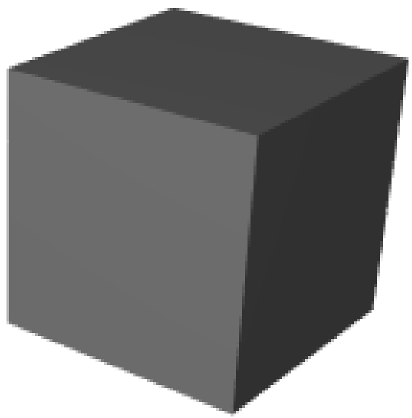
Tutaj postawiłem źródło światła w podanym punkcie i zdefiniowałem jego kolor na `<1,1,1>` w modelu kolorów RGB, czyli emitowane światło jest białe.

I teraz zostało nam już tylko po trzecie, czyli obserwator. Program musi wiedzieć z której strony i z jakiej odległości chcemy obserwować obraz.

```
camera {location <6,5,-7> look_at <0,0,0>}
```

Teraz zdefiniowałem punkt obserwacji i kierunek wskazujący na środek układu współrzędnych.

I to już wszystko. Teraz wystarczy uruchomić program i zobaczymy nasz sześcian.



Autorami rysunków zamieszczonych w tekście są: M. Miller, B. Jackowski, oraz autor niniejszego artykułu. Wszystkie wygenerowane zostały za pomocą programów POV i Vivid.