

Przekształcenia afiniczne

We współrzędnych kartezjańskich:

$$f(\mathbf{p}) = L\mathbf{p} + \mathbf{t},$$

L oznacza macierz 3×3 części liniowej, \mathbf{t} to wektor przesunięcia.

We współrzędnych jednorodnych przekształcenie jest dokonywane za pomocą mnożenia przez macierz

$$A = \left[\begin{array}{c|c} L & \mathbf{t} \\ \hline 0 & 0 \ 0 \ 1 \end{array} \right],$$

zatem wektor $A\mathbf{p}$ reprezentuje punkt $f(\mathbf{p})$.

Obie reprezentacje przekształceń są związane z konkretnym układem współrzędnych kartezjańskich.

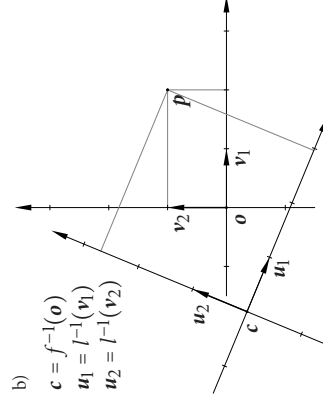
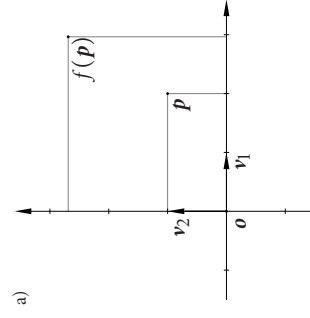
1

Interpretacja „zwykła” i dualna

Wzór opisujący przekształcenie f można interpretować na dwa sposoby:

- Wektor $A\mathbf{p}$ + \mathbf{t} składa się ze współrzędnych kartezjańskich nowego punktu.
- Wektor $A\mathbf{p}$ + \mathbf{t} składa się ze współrzędnych kartezjańskich tego samego punktu w nowym układzie współrzędnych.

2



Przekształcenie afiniczne i jego dualna interpretacja

3

Macierz przekształcenia afinicznego f można przedstawić w postaci

$$A = \begin{bmatrix} w_1 & w_2 & w_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Zatem jej kolumny są wektorami współrzędnych jednorodnych trzech wektorów swobodnych i jednego punktu.

Jeśli wektory w_1, w_2, w_3 są liniowo niezależne, to razem z punktem \mathbf{t} określają układ odniesienia dla nowego układu współrzędnych kartezjańskich.

Punkt \mathbf{t} jest obrazem początku \mathbf{o} bieżącego układu współrzędnych w przekształceniu f , a wektor w_i jest obrazem wektora osi e_i w przekształceniu liniowym f opisanym przez macierz $L = [w_1, w_2, w_3]$.

4

W interpretacji dualnej: niech $\mathbf{c} = f^{-1}(\mathbf{o})$, $\mathbf{u}_1 = l^{-1}(\mathbf{v}_1)$, $\mathbf{u}_2 = l^{-1}(\mathbf{v}_2)$, $\mathbf{u}_3 = l^{-1}(\mathbf{v}_3)$. Punkt \mathbf{c} i wektory $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ stanowią układ odniesienia dla układu współrzędnych, do którego przeszliśmy, tj. $L\mathbf{p} + \mathbf{t}$ jest wektorem współrzędnych kartezjańskich danego punktu w tym nowym układzie.

5

Składanie przekształceń afinicznych

Niech A_1 i A_2 będą macierzami przekształceń f_1 i f_2 , wykonanych kolejno.

Jeśli oba przekształcenia są określone w tym samym układzie współrzędnych, to złożenie $f = f_2 \circ f_1$, tj. przekształcenie dane wzorem $f(\mathbf{p}) = f_2(f_1(\mathbf{p}))$ jest reprezentowane przez macierz A_2A_1 .

Jeśli przekształcenia podajemy w układzie związanym z przekształcanym przedmiotem, który np. obraca się razem z tym przedmiotem, to w układzie początkowym przekształcenie f_1 jest określone przez macierz A_1 , a macierz przekształcenia f_2 w tym układzie jest równa $A_1A_2A_1^{-1}$.

Kolejno: A_1^{-1} opisuje przejście do układu zmienionego przez f_1 , A_2 reprezentuje przekształcenie f_2 w tym układzie, A_1 opisuje powrót do układu początkowego. Złożenie tak określonych przekształceń jest opisane przez macierz

$$A_1A_2A_1^{-1}A_1 = A_1A_2.$$

6

Pierwszy sposób składania przekształceń przydaje się, gdy obracamy przedmiot, aby go oglądać z różnych stron.

*Za czym spod ich ręk wypelzła szara myszka cynowa,
która puszczając pyszczkiem bałki mydlane, zarazem
biegła po stole, a spod ogonka sypał się jej biały kredowy
pył tak kunsztownie, iż powstał z tego kaligrafowany napis:
A WIĘC NAPRAWDĘ NAS NIE KOCHACIE?*

Stanisław Lem: *Cyberiada*

Drugi sposób jest stosowany w tzw. **grafice żółwia**. Żółw chodzi po płaszczyźnie, wykonując polecenia w rodzaju „Obróć się o x stopni w lewo” lub „Wykonaj krok o długości y w przód”.

7

Przekształcanie wektora normalnego

Wektor normalny powierzchni jest jednym z najważniejszych elementów modelu oświetlenia. Dowolny obiekt może być opisany w takim układzie współrzędnych, jaki daje projektantowi największą wygodę, ale potem trzeba wszystkie obiekty przekształcić tak, aby były odpowiednio rozmieszczone względem siebie, czyli dla każdego obiektu przejść do wyróżnionego układu współrzędnych świata.

Punkty przekształcamy bezpośrednio, wzorem $f(\mathbf{p}) = L\mathbf{p} + \mathbf{t}$.

Wektory swobodne są różnicami punktów: jeśli $\mathbf{v} = \mathbf{p}_1 - \mathbf{p}_2$, to obrazem \mathbf{v} jest

$$f(\mathbf{p}_1) - f(\mathbf{p}_2) = L\mathbf{p}_1 + \mathbf{t} - (L\mathbf{p}_2 + \mathbf{t}) = L(\mathbf{p}_1 - \mathbf{p}_2) = L\mathbf{v}.$$

8

Wektor normalny powierzchni nie jest wektorem swobodnym. Sposób jego przekształcania wynika z równania płaszczyzny: $\pi = \{ \mathbf{p} : \mathbf{n}^T(\mathbf{p} - \mathbf{p}_0) = 0 \}$.

Równanie obrazu płaszczyzny π w przekształceniu f to

$$0 = \mathbf{m}^T (f(\mathbf{p}) - f(\mathbf{p}_0)) = \mathbf{m}^T L(\mathbf{p} - \mathbf{p}_0).$$

Równanie będzie spełnione, jeśli przyjmiemy $\mathbf{m} = L^{-T} \mathbf{n}$, bo wtedy

$$\mathbf{n}^T L^{-1} L(\mathbf{p} - \mathbf{p}_0) = \mathbf{n}^T (\mathbf{p} - \mathbf{p}_0) = 0.$$

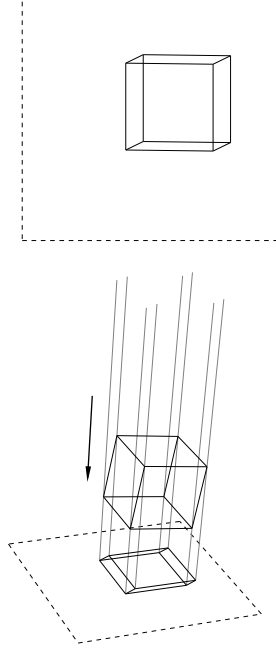
Zauważmy, że $L^{-T} = L$ wtedy (i tylko wtedy), gdy macierz L jest ortogonalna, czyli gdy przekształcenie f jest izometrią.

W ogólności macierzy L można użyć do przekształcania wektora normalnego, gdy przekształcenie f jest podobieństwem geometrycznym. Długość wektora normalnego na ogół nie jest istotna, w zastosowaniach zazwyczaj wektor ten normalizujemy, otrzymując wektor jednostkowy o tym samym kierunku i zwrocie.

9

Rzutowanie

Rzutowanie równoległe



Szczególnymi przypadkami rzutowania równoległego są **rzutowanie prostopadłe** i **aksonometria**, najczęściej stosowane w rysunku technicznym.

10

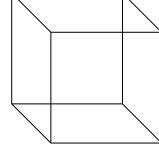
Aksonometria jest zazwyczaj rzutem ukosnym, w którym kierunek rzutowania nie jest prostopadły do rzutni.

Twierdzenie Pohlkego orzeka, że dla dowolnej czwórki punktów $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ nieleżących w jednej płaszczyźnie i dowolnej czwórki punktów $(\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$ na płaszczyźnie, takich że żadne trzy nie są współliniowe, istnieje rzutnia i kierunek rzutowania, w którym obraz czwórki $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ jest figurą podobną do czwórki $(\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$.

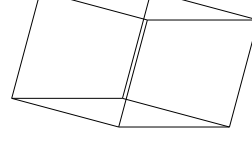
Na tej podstawie można (prawie) dowolnie wybierać obrazy początku układu współrzędnych i jego wersorów osi.

11

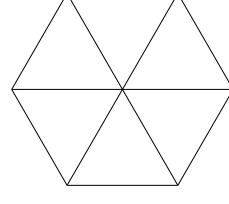
Najczęściej spotykane w rysunku technicznym rodzaje aksonometrii:



kawalerska



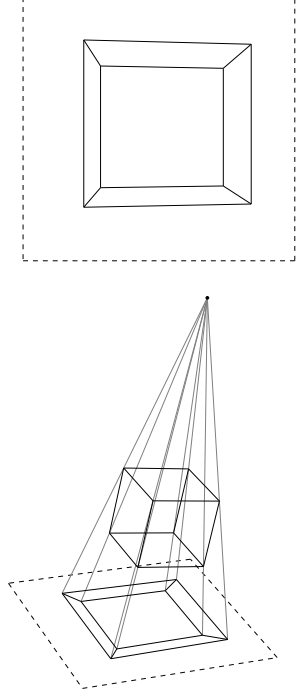
wojskowa



izometryczna

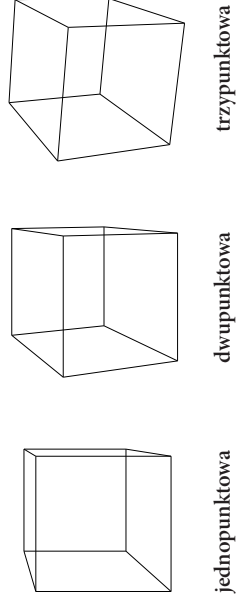
12

Rzutowanie perspektywiczne



13

Perspektywa



14

Rzutowanie w OpenGL-u

Przekształcenia, którym poddawane są wierzchołki w celu otrzymania obrazu, są kolejnymi zmianami układów współrzędnych.

- **Układ współrzędnych modelu** jest to układ, w którym podawane są na wejście potoku przetwarzania grafiki współrzędne położenia wierzchołków.
Dla każdego obiektu wchodzącego w skład rysowanej sceny może być inny układ współrzędnych modelu.
- **Układ współrzędnych świata** jest to jeden wspólny układ współrzędnych, do którego najpierw dokonuje się przejścia od układu (układów) współrzędnych modelu.

W układzie współrzędnych świata zazwyczaj oblicza się oświetlenie, cienie i kolizje obiektów.

15

- **Układ współrzędnych obserwatora** określa usytuowanie „kamery” względem rysowanej sceny.

Najczęściej przejście od układu świata do układu obserwatora jest izometrią, która umieszcza w świetle środek rzutowania perspektywicznego i określa kierunek, w którym ten obserwator patrzy.

- **Układ współrzędnych kostki standardowej** (*normalized device coordinates*, *NDC*) nie jest kartezjański, jeśli rzutowanie nie jest równoległe (czyli np. jest perspektywiczne).

Kostka standardowa jest to sześcián $[-1, 1]^3$; jest to obszar, którego zawartość może być widoczna na obrazie — wszystko, co wystaje poza nią zostanie obcięte.

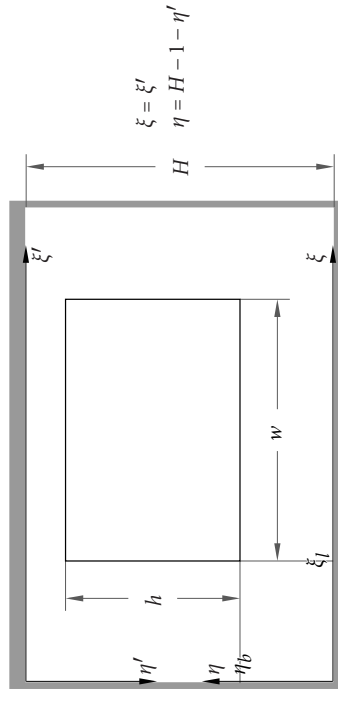
Przejście od układu obserwatora do układu kostki standardowej, jeśli rzutowanie jest perspektywiczne, jest przekształceniem rzutowym, reprezentowanym przez macierz 4×4 .

16

- Układ współrzędnych klatki OpenGL-a jest to układ określony w pewnym prostokącie na ekranie. Jednostki osi tego układu są równe szerokości i wysokości jednego piksela.
- Układy współrzędnych okna OpenGL-a i systemu okien są związane z oknem, w którym jest wyświetlana grafika.
Najczęściej (chyba zawsze) układy klatki OpenGL-a i okna mają osie y (dalej oznaczam η i η') zorientowane przeciwnie i początku w innych narożnikach okna.

17

Klatka w oknie



18

Aby określić klatkę o wskazanych wymiarach i położeniu w oknie, trzeba wykonać instrukcję

```
glViewport ( xi_l, eta_b, w, h );
```

albo

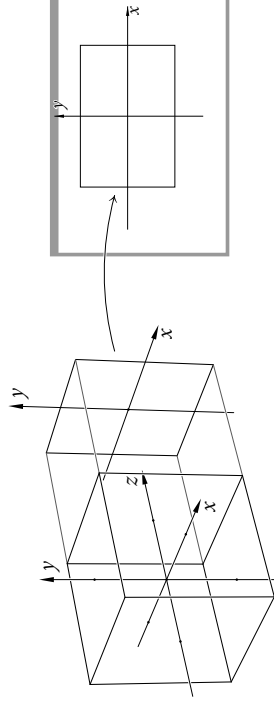
```
glViewport ( xiprime_l, H-h-etaprime_t, w, h );
```

Jeśli klatka ma być całym oknem, to piszemy w programie

```
glViewport ( 0, 0, W, H );
```

19

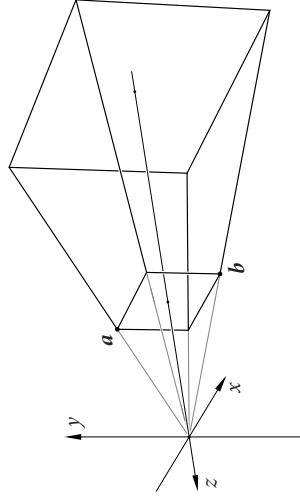
Odwzorowanie kostki standardowej na klatkę



To przekształcenie jest wykonywane między etapem obcinania a etapem rasteryzacji. Proste o kierunku osi z są odwzorowane na punkty. Z dwóch punktów na takiej prostej położony „bliziej obserwatora” jest ten, który ma mniejszą współrzędną z w układzie kostki standardowej.

20

Implementacja rzutowania perspektywicznego



Bryła widzenia jest ściętym ostrosłupem, określonym przez podanie sześciu liczb — l, r, b, t, n, f . Punkty $\mathbf{a} = (l, t, -n)$ i $\mathbf{b} = (r, b, -n)$ są wierzchołkami ostrosłupa, którego tylna ściana leży w płaszczyźnie $z = -f$.

21

Litery w oznaczeniach tych sześciu parametrów są początkami słów **l**eft, **r**ight, **b**ottom, **t**op, **n**ear, **f**ar.

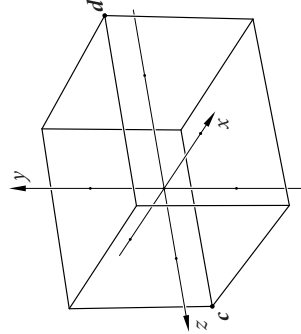
Macierz P , reprezentująca przekształcenie (kolineację rzutową) określoną tak, aby obrazem bryły widzenia była kostka standardowa, oraz jej odwrotność są dane wzorami

$$P = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2fn}{n-f} \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad P^{-1} = \begin{bmatrix} \frac{r-l}{2n} & 0 & 0 & 0 \\ 0 & \frac{t-b}{2n} & 0 & 0 \\ 0 & 0 & \frac{n-f}{2fn} & \frac{n+f}{2fn} \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Przez macierz P mnożymy wektor współrzędnych jednorodnych punktu, otrzymując wektor współrzędnych jednorodnych w układzie kostki standardowej. Na ogół współrzędna wagowa tego wektora będzie różna od 1.

22

Dla rzutowania równoległego bryła widzenia jest (w układzie obserwatora) prostopadłościąn.



Jest on określony przez punkty $\mathbf{c} = (l, b, -n)$ i $\mathbf{d} = (r, t, -f)$.

23

Macierz P , przekształcająca bryłę widzenia na kostkę standardową i odwrotność tej macierzy są dane wzorami

$$P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & \frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & \frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{n-f} & \frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P^{-1} = \begin{bmatrix} \frac{r-l}{2} & 0 & 0 & \frac{l+r}{2} \\ 0 & \frac{t-b}{2} & 0 & \frac{b+t}{2} \\ 0 & 0 & \frac{n-f}{2} & \frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

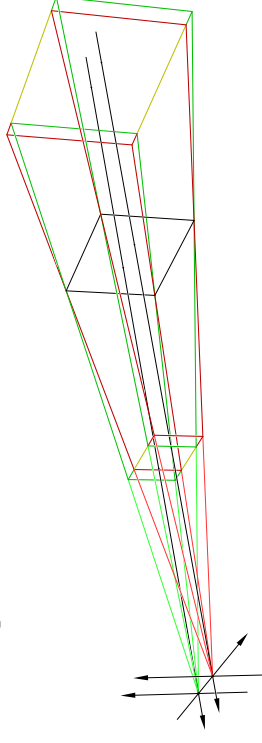
24

Każde urządzenie rastrowe ma określony tzw. **aspekt** — jest to iloraz szerokości i wysokości piksela. Klatka o wymiarach $w \times h$ pikseli ma fizyczne wymiary o proporcji $aw : h$. Aby uniknąć zniekształceń obiektów, podczas określania bryły widzenia (w obu przypadkach) trzeba zapewnić, że $(r - l) : (t - b) = aw : h$.

Obecnie najczęściej aspekt $a = 1$ lub jest to liczba, której zastąpienie przez 1 daje błąd zaniedbywalny.

25

Stereoskopia



Dwa rzuty perspektywiczne dla stereoskopii wymagają wzięcia pod uwagę czterech wymiarów fizycznych: **roztawu oczu obserwatora, odległości obserwatora od płaszczyzny ekranu oraz szerokości i wysokości klatki.**

26

Inaczej niż w „zwykłym” rzutowaniu perspektywicznym, trzeba jednostkę długości układu współrzędnych obserwatora powiązać z wymiarami fizycznymi, wyrażanymi w milimetrach — albo, w drugą stronę, trzeba wymienione wyżej wymiary fizyczne wyrazić w jednostkach układu obserwatora, aby na tej podstawie ustalić parametry l, r, b, t, n, f wyznaczające bryły (ostroslupy) widzenia. Dla obu ostrosłupów parametry b, t, n i f będą wspólne.

Najwygodniej jest przyjąć, że przejście między układami współrzędnych świata i obserwatora jest zawsze izometrią — wtedy jednostki długości w obu układach są takie same.

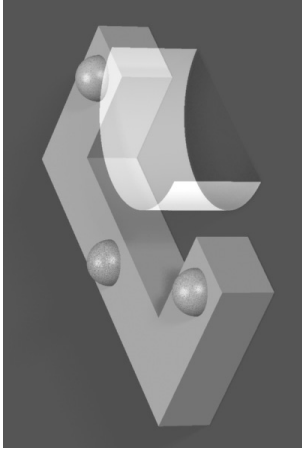
27

Rzuty nieliniowe

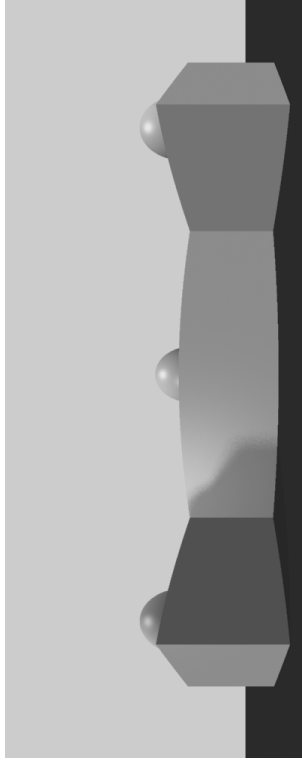
Inne niż rzuty równoległe i perspektywiczne odwzorowania przestrzeni trójwymiarowej na płaszczyznę nie zachowują współliniowości punktów — obrazy odcinków mogą być zakrzywione, co utrudnia ich rysowanie. Jeśli trzeba wykonać obraz w takim rzucie, to albo należy użyć podprogramów rysujących odpowiednio krzywe, albo rozdrobnić odcinki i trójkąty, zastępując je łamanymi lub powierzchniami zbudowanymi z dostatecznie małych trójkątków.

28

Najczęściej spotykanym odwzorowaniem nieliniowym jest **panorama**, czyli rzut środkowy na rzutnię walcową, następnie rozwijaną. Środek rzutowania znajduje się na osi walca.

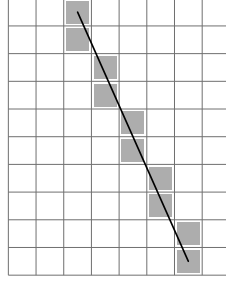


29



30

Algorytmy rasteryzacji



Dla odcinka należy wyznaczyć piksele, z których powstanie jego obraz. Wprowadzimy klasyczny algorytm **Bresenhama**.

31

Założymy, że końce odcinka mają współrzędne całkowite (tj. pokrywają się ze środkami pikseli), (x_0, y_0) i (x_1, y_1) . Chwilowo przyjmujemy $x_1 > x_0, y_1 \geq y_0$.

W każdej kolumnie rastra od x_0 do x_1 wybierzemy jeden piksel, pokoloruje go jakaś procedura **SetPixel**.

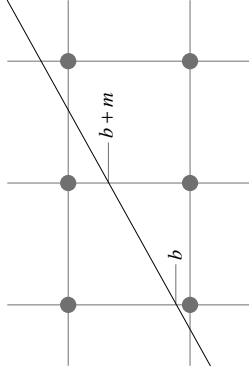
Pierwszy algorytm:

```
 $\Delta x = x_1 - x_0; \quad \Delta y = y_1 - y_0; \quad m = \Delta y / \Delta x;$   
for (  $x = x_0, y = y_0; x \leq x_1; x++, y += m$  )  
    SetPixel (  $x, \text{round}(y)$  );
```

Zmiennym m oraz y przyjmują wartości ułamkowe, zatem trzeba używać arytmetyki zmiennopozycyjnej — jest kosztowniejsza niż stałopozycyjna i są w niej błędy zaokrążeń. Ponadto trzeba zaokrąślać zmienną y .

32

W każdej kolejnej kolumnie albo pozostajemy w poprzednim wierszu, albo idziemy o 1 wiersz wyżej — to zależy, środek którego piksela jest bliższy rysowanego odcinka, czyli daje **mniej** błąd.



Tu kropki oznaczają środki pikseli. Litera b oznacza odległość punktu przecięcia odcinka z linią pionową rastra od środka piksela (czyli błąd). W następnej kolumnie błąd jest równy $b + m$, jeśli jest on większy niż $1/2$, to przejdziemy do następnego wiersza.

Drugi algorytm:

```

 $\Delta x = x_1 - x_0; \quad \Delta y = y_1 - y_0; \quad m = \Delta y / \Delta x;$ 
for (  $x = x_0, y = y_0, b = 0; \quad x \leq x_1; \quad x++$  ) {
    SetPixel (  $x, y$  );
     $b += m;$ 
    if (  $b > 0.5$  ) {  $y++; \quad b -= 1.0;$  }
}

```

Teraz zmienna y przyjmuje tylko wartości całkowite, ale zmienne m i b mogą mieć wartości ułamkowe.

Wszystkie ułamki, które mogą być wartościami tych zmiennych są ilorazami liczb całkowitych, z mianownikiem $2\Delta x$. Możemy więc prowadzić rachunki tylko na licznikach tych ułamków.

Algorytm Bresenhama:

```

 $\Delta x = x_1 - x_0; \quad \Delta y = y_1 - y_0;$ 
for (  $x = x_0, y = y_0, c = -\Delta x; \quad x \leq x_1; \quad x++$  ) {
    SetPixel (  $x, y$  );
     $c += 2\Delta y;$ 
    if (  $c > 0$  ) {  $y++; \quad c -= 2\Delta x;$  }
}

```

Tu wszystkie działania są wykonywane na liczbach całkowitych. Jeśli nie jest spełnione ograniczenie $x_1 > x_0, y_1 \geq y_0$, to trzeba zamienić współrzędne x, y rolami i odpowiednio dobrać znaki przyrostów zmiennych.

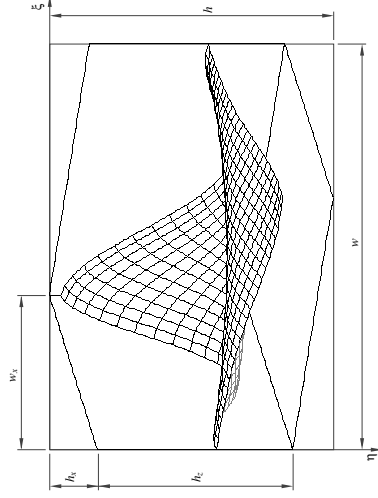
```

 $dx = x_1 - x_0; \quad g = dx > 0 ? +1 : -1; \quad dx = abs(dx);$ 
 $dy = y_1 - y_0; \quad h = dy > 0 ? +1 : -1; \quad dy = abs(dy);$ 
if (  $dx > dy$  ) {
    for (  $x = x_0, y = y_0, c = -dx; \quad x != x_1; \quad x += g$  ) {
        SetPixel (  $x, y$  );
         $c += 2*dy;$ 
        if (  $c > 0$  ) {  $y += h; \quad c -= 2*dx;$  }
    }
}
else {
    for (  $x = x_0, y = y_0, c = -dy; \quad y != y_1; \quad y += h$  ) {
        SetPixel (  $x, y$  );
         $c += 2*dx;$ 
        if (  $c > 0$  ) {  $x += g; \quad c -= 2*dy;$  }
    }
}

```

Algorytm z pływającym horyzontem

Wykres funkcji dwóch zmiennych w rzucie aksjonometrycznym z „widocznością”:



37

Narysujemy siatkę linii na powierzchni $z = f(x, y)$ dla funkcji ciągłej

$$f: [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \rightarrow [z_{\min}, z_{\max}] \subset \mathbb{R}.$$

Aby skonstruować rzut aksjonometryczny, trzeba podać na przykład wymiary w, h prostokąta otaczającego wykres i wymiary w_x, h_x i h_z . Obrazami punktów

$(x_{\min}, y_{\min}, z_{\min}), (x_{\min}, y_{\min}, z_{\max}), (x_{\max}, y_{\min}, z_{\min}), (x_{\max}, y_{\max}, z_{\min})$ będą punkty $(w_x, h_x), (w_x, 0), (0, h_x + h_z)$ i $(w - w_x, h)$. Na tej podstawie łatwo jest znaleźć współczynniki a_ξ, \dots, d_ξ i a_η, \dots, d_η występujące we wzorach

$$\xi = a_\xi x + b_\xi y + c_\xi z + d_\xi,$$

$$\eta = a_\eta x + b_\eta y + c_\eta z + d_\eta.$$

38

Rysowanie jest wykonywane „od przodu do tyłu”. Podczas rysowania odcinka odcinki narysowane wcześniej określają „obszar zasłonięty”. Obszar ten jest reprezentowany przez dwie tablice liczb całkowitych zwane **horyzontami** **hb** i **ht**. Każdy element tablic odpowiada kolumnie pikseli.

Zgodnie z orientacją osi η , wszystkim elementom horyzontu dolnego (**hb**) jest przypisywana wartość początkowa -1 , a górnego (**ht**) $h + 1$; wartości elementów dolnego horyzontu będą rosłać, a górnego maleć. W kolumnie x obszar zabroniony do rysowania jest między wierszami **ht** [x] i **hb** [x] (jeśli **ht** [x] $>$ **hb** [x]), to obszar zabroniony jest pusty.

39

Jeśli piksel (x, y) rysowanego odcinka spełnia warunek **ht** [x] $\leq y \leq$ **hb** [x], to znajduje się w obszarze zabronionym i należy go pominąć. W przeciwnym razie jest albo $y <$ **ht** [x] (piksel jest nad górnym horyzontem) albo $y >$ **hb** [x] (piksel jest poniżej dolnego horyzontu). Należy go narysować (można uzależnić kolor od tego, która z tych nierówności jest spełniona) i przypisać współrzędną y odpowiedniemu horyzontowi (albo obu, jeśli to jest pierwszy rysowany piksel w kolumnie x).

40

Problemem jest uniknięcie sytuacji, w której piksele odcinka zasłaniają inne piksele tego samego odcinka, znajdujące się w tej samej kolumnie i jeszcze nie narysowane.

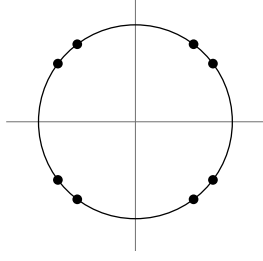
Aby sobie z tym poradzić, odcinki trzeba rysować dwukrotnie. Za pierwszym razem sprawdzamy nierówności i rysujemy widoczne piksele, ale nie zmieniamy zawartości horyzontów. Za drugim razem uaktualniamy horyzonty, już bez rysowania.

Całkowicie poprawny efekt uzyskamy, rysując odcinki parami (tj. najpierw rysujemy dwa mające wspólny koniec odcinki — obrazy odcinków równoległych do płaszczyzn xz i yz , a następnie uaktualniamy horyzonty dla obu tych odcinków.

41

Rasteryzacja okręgu

Rachunki potrzebne do narysowania okręgu, którego promień r i współrzędne środka są liczbami całkowitymi również można sprowadzić do działań stałopozycyjnych. Jeśli środkiem jest punkt $(0, 0)$, to mając jeden piksel, (x, y) , możemy podać 7 innych pikseli, zamieniając współrzędne i nadając im wszystkie kombinacje znaków.

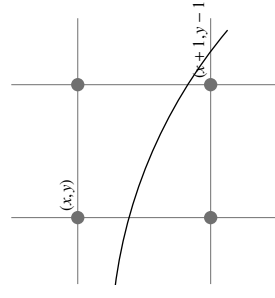


42

Będziemy wyznaczać piksele (x, y) , takie że $0 \leq x \leq y \leq r$, otrzymując pozostałe z symetrii. Wystartujemy od piksela $(0, r)$ i będziemy w każdym kroku zwiększać x i w pewnych krokach zmniejszać y .

Wartość funkcji f przyjmujemy za miarę błędu; po narysowaniu piksela (x, y) współrzędną y zmniejszymy, jeśli

$$|f(x, y)| < |f(x, y + 1)|.$$



44

Rozważmy dwie tożsamości:

$$\sum_{i=0}^x (2i + 1) = (x + 1)^2,$$

$$\sum_{i=y}^r (2i - 1) = r^2 - (y - 1)^2.$$

Funkcja

$$f(x, y) = \sum_{i=0}^x (2i + 1) - \sum_{i=y}^r (2i - 1) = (x + 1)^2 + (y - 1)^2 - r^2$$

ma wartość 0, jeśli punkt $(x + 1, y - 1)$ leży na okręgu, jest dodatnia jeśli na zewnątrz i ujemna jeśli leży wewnątrz.

43

Niech $c = f(x, y)$. Wtedy
 $f(x, y+1) = c + 2y - 1, f(x+1, y) = c + 2x + 3, f(x+1, y-1) = f(x+1, y) - 2y + 3.$

Podczas rysowania zawsze wybieramy między pikselem wewnątrz i na zewnątrz okręgu, czyli mamy $f(x, y) \leq 0 \leq f(x, y+1)$. To umożliwia uniknięcie obliczania wartości bezwzględnych: warunek równoważny nierówności

$|f(x, y)| < |f(x, y+1)|$ to $-c < c + 2y - 1$, czyli $2c > 1 - 2y$. Stąd algorytm:

```
for ( x = 0, y = r, c = 2*(1-r); x <= y; x++, c += 2*x+1 )
{
    SetPixels ( x, y );
    if ( 2*c > 1-2*y ) {
        y --;
        c += 1-2*y;
    }
}
```

45

Oczywiście, na ekranie zobaczymy okrąg, jeśli współczynnik aspekt tego ekranu jest równy (lub dostatecznie bliski) 1. W przeciwnym razie trzeba by narysować elipsę, której osie główne są równoległe do krawędzi ekranu i których długości są odpowiednio dobrane.

Można narysować elipsę, której półosie — pozioma i pionowa — mają długości a i b . Algorytm, który to robi, polega na narysowaniu okręgu o promieniu $r = ab$, przy czym raster, na którym to się odbywa jest wirtualny. Współrzędną x na takim rasterze zwiększamy o b , a współrzędną y zmniejszamy o a . To oznacza konieczność dodawania do i odejmowania od zmiennej c kolejnych składników w sumach użytych do zdefiniowania funkcji f . Oczywiście, po „fizycznym” rasterze poruszamy się w obu kierunkach z krokiem 1.

46

Niech $c = f(x, y)$. Wtedy

$$f(x, y+1) = c + 2y - 1, f(x+1, y) = c + 2x + 3, f(x+1, y-1) = f(x+1, y) - 2y + 3.$$

Podczas rysowania zawsze wybieramy między pikselem wewnątrz i na zewnątrz okręgu, czyli mamy $f(x, y) \leq 0 \leq f(x, y+1)$. To umożliwia uniknięcie obliczania wartości bezwzględnych: warunek równoważny nierówności

$|f(x, y)| < |f(x, y+1)|$ to $-c < c + 2y - 1$, czyli $2c > 1 - 2y$. Stąd algorytm:

```
for ( x = 0, y = r, c = 2*(1-r); x <= y; x++, c += 2*x+1 )
{
    SetPixels ( x, y );
    if ( 2*c > 1-2*y ) {
        y --;
        c += 1-2*y;
    }
}
```

45

Zatem, bierzemy $r = ab$ i określamy funkcję

$$f(x, y) = (x+b)^2 + (y-a)^2 - r^2 = \sum_{i=0}^{x+b-1} (2i+1) - \sum_{i=y-a+1}^r (2i-1).$$

Mamy stąd

$$\begin{aligned} f(x, y+a) &= f(x, y) + (2y-a)a, \\ f(x+b, y) &= f(x, y) + (2x+3b)b, \\ f(x+b, y-a) &= f(x+b, y) + (3a-2y)a. \end{aligned}$$

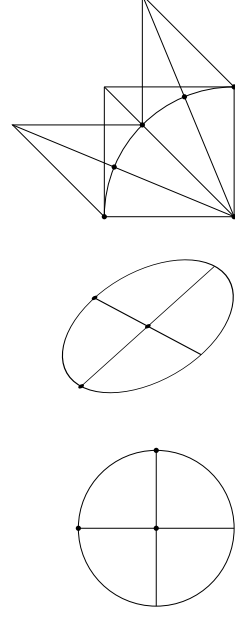
W chwili rysowania piksela (x, y) , mamy obliczone $c = f(x, y)$. Idąc z krokiem b w lewo, decydujemy, czy iść też z krokiem a w dół. W tym celu porównujemy $|f(x, y)|$ i $|f(x, y+b)|$, czyli równoważnie sprawdzamy warunek $-c < c + (2y-a)a$, czyli $2c + (2y-a)a > 0$ — jeśli jest spełniony, to idziemy także w dół, odpowiednio uaktualniając wartość zmiennej c .

Obraz elipsy ma tylko czterokrotną symetrię, a można narysować tylko 1/8 łuku, dla $x \leq y$. Drugą połowę ćwiartki łuku (dla $x > y$) trzeba narysować osobno, w podobny sposób.

47

Inny sposób rysowania krzywych (w tym elips) polega na obliczeniu ciągu punktów na krzywej i zastąpienia tej krzywej przez łamaną (a potem narysowanie odcinków). Gęstość punktów na krzywej można dobrać adaptacyjnie, aby osiągnąć wymaganą dokładność możliwie małym kosztem.

Elipsę w położeniu ogólnym można reprezentować przez podanie środka i wektorów **półosi sprzężonych**. Osie sprzężone są obrazami średnic prostopadłych okręgu w przekształceniu afinicznym przeprowadzającym ten okrąg na daną elipsę.



48

Niech v_0 i v_1 oznaczają wektory wyznaczające końce łuku okręgu zajmującego kąt α . Wektor wyznaczający punkt w połowie tego łuku jest równy $v = (v_0 + v_1)/c$, gdzie $c = 2 \cos \alpha/2$. Możemy przyjąć $c_i = \cos \pi/2^{i+1}$ i obliczyć (w preprocesingu) kilka(naście) początkowych wyrazów ciągu $1/c_0, \dots, 1/c_{n-1}$. To umożliwi rekurencyjne dzielenie łuku okręgu na połowy, ćwiartki itd.

Zamiast wzajemnie prostopadłych wektorów o tej samej długości (półosi sprzężonych okręgu) możemy na początku rekurencji przyjąć dwa *dowolne* wektory — odpowiadające półosiom sprzężonym elipsy. Wtedy kolejne punkty podziału będą punktami tej elipsy.

Zatrzymanie rekurencji może nastąpić z powodu osiągnięcia maksymalnej głębokości lub po otrzymaniu dostatecznie krótkiego odcinka.

49

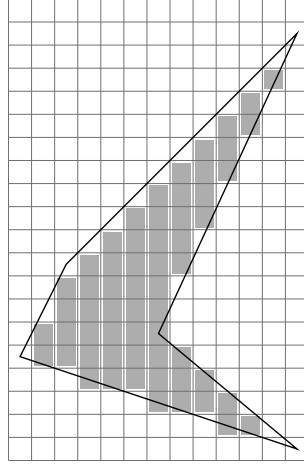
Wersja rekurencyjna może być mniej więcej taka:

```
void r_Elipsa ( k, s, v0, v1 )
{
    if ( k == n || dostatecznie_blisko ( v0, v1 ) )
        rysuj_odcinek ( s+v0, s+v1 );
    else {
        v2 = a[k]*(v0+v1);
        r_Elipsa ( k+1, s, v0, v2 );
        r_Elipsa ( k+1, s, v2, v1 );
    }
} /*r_Elipsa*/
```

Pierwszy parametr wywołania w aplikacji ma być równy 0.

50

Wypełnianie wielokątów



Dobrze zaprojektowana procedura rasteryzacji powinna zapewniać jednoznaczność „zaklasyfikowania” pikseli do wielokątów o wspólnych bokach. Do obrazu wielokąta należą te piksele, których środki są wewnątrz. Co z pikselami na brzegu?

51

Przykładowo:

Jeśli bezpośrednio na prawo od środka piksela są punkty należące do *wewnątrz* wielokąta, to ten piksel należy do obrazu wielokąta.

Jeśli na prawo od środka piksela jest krawędź pozioma, to piksel należy do obrazu wielokąta, gdy bezpośrednio poniżej jest wewnątrz lub lewa krawędź pionowa wielokąta.

Dzięki takiej (lub podobnej) umowie płaszczyzna pokryta wielokątami mającymi tylko wspólne krawędzie lub wierzchołki zostanie pomalowana poprawnie — każdy piksel otrzyma kolor i to tylko raz.

Standard OpenGL nie narzuca konkretnej umowy dla implementacji.

52

Reguła parzystości: punkt nieleżący na brzegu (ograniczonego) wielokąta jest w jego wnętrzu, jeśli dowolna półprosta wychodząca z tego punktu przecina brzeg w nieparzystej liczbie punktów.

Sekwencyjny algorytm wypełniania wielokąta poziomymi odcinkami używa **tablicy krawędzi aktywnych**. Krawędź jest aktywna, jeśli nie jest pozioma i dla ustalonego y ma punkt wspólny z prostą poziomą na wysokości y . Dla każdej krawędzi niepoziomej odrzucamy jej dolny koniec.

Zgodnie z regułą parzystości liczba krawędzi aktywnych jest zawsze parzysta.

53

Algorytm: (zakładam, że wierzchołki mają współrzędne całkowite)

- Na podstawie tablicy wierzchołków utwórz tablicę krawędzi (par kolejnych wierzchołków), w tym krawędź „zamykającą” $(x_{n-1}, y_{n-1})(x_0, y_0)$. Pomiń wszystkie krawędzie poziome.
- Jeśli dla dowolnej krawędzi $(x_i, y_i)(x_{i+1}, y_{i+1})$ jest $y_{i+1} < y_i$, to przestaw końce tej krawędzi.
- Posortuj krawędzie w tablicy w kolejności rosnących współrzędnych y pierwszego wierzchołka.
- Utwórz początkowo pustą tablicę krawędzi aktywnych.

54

- W pętli, dla zmiennej y przebiegającej od minimalnej do maksymalnej współrzędnej y linii rastra mających przecięcie z wielokątem.

- Usuń z tablicy krawędzi aktywnych wszystkie krawędzie, których drugi koniec jest na wysokości y .
- Dołącz do tablicy krawędzi aktywnych wszystkie krawędzie, których pierwszy koniec jest na wysokości y .
- Oblicz współrzedną x przecięcia każdej krawędzi aktywnej z prostą poziomą na wysokości y .
- Posortuj tablicę krawędzi aktywnych w kolejności rosnących współrzędnych x punktów przecięcia.
- Wypełnij kolorem poziome odcinki na wysokości y między kolejnymi parami punktów przecięcia.

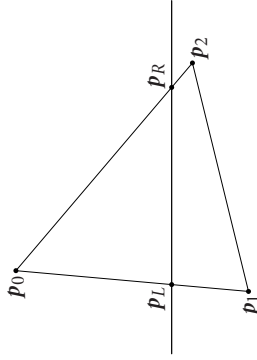
55

Algorytm rasteryzacji w implementacjach OpenGL-a dopuszcza tylko trójkąty, a ponadto dopuszcza ulamkowe (w reprezentacji zmiennopozycyjnej) współrzędne wierzchołków. Ponadto algorytm pracuje we współrzędnych jednorodnych, obliczając dla każdego piksela (równoległe) współrzedną z w układzie kostki standardowej. Jest ona potrzebna do testów widoczności i jest też przekazywana na wejście szadera fragmentów, który może jej użyć w dowolnym obliczeniu, a także zmienić (co ma wpływ na wyniki testu widoczności, wykonywanego później).

Algorytm rasteryzacji OpenGL-a dokonuje także interpolacji wszystkich dodatkowych atrybutów wierzchołków trójkąta. Algorytm jest równoległy.

56

Wierzchołki trójkąta są reprezentowane przez wektory współrzędnych jednorodnych, $P_i = (X_i, Y_i, Z_i, W_i)$. Wszystkie wagi mają ten sam znak. Najpierw trzeba znaleźć wierzchołki, które mają najmniejszą i największą współrzędną y .



Dla poziomej linii rastra na wysokości y trzeba znaleźć punkty P_L i P_R — przecięcia boków trójkąta z tą linią, przy czym najpierw trzeba ustalić, które to są boki. Dalej przyjmijmy, że jest tak jak na rysunku.

57

Parametryczne przedstawienie odcinka $\overline{P_0P_1}$ w reprezentacji jednorodnej:

$$\{ P: P = (1-t)P_0 + tP_1, t \in [0, 1] \}.$$

Poszukujemy takiego punktu $P_L = (X, Y, Z, W)$, dla którego $Y/W = y$. Z równości

$$\frac{(1-t)Y_0 + tY_1}{(1-t)W_0 + tW_1} = y$$

wynika

$$t = \frac{-Y_0 - W_0y}{(Y_1 - Y_0) - (W_1 - W_0)y} \quad \text{oraz} \quad x_L = \frac{X_L}{W_L} = \frac{(1-t)X_0 + tX_1}{(1-t)W_0 + tW_1}.$$

Podobnie znajdujemy punkt P_R i współrzędną x_R punktu P_R . To można zrobić równoległe dla wszystkich poziomych linii rastra przeciętych przez trójkąt.

58

Mając dla ustalonego y punkty P_L i P_R , można dla równoległe dla każdego piksela (x, y) odcinka $\overline{P_LP_R}$ obliczyć punkt P trójkąta, odwzorowany na ten piksel. Jest

$$P = (1-s)P_L + sP_R, \quad s = \frac{-X_L - W_Lx}{(X_R - X_L) - (W_R - W_L)x}$$

i dalej analogicznie jak w poprzednim kroku. W szczególności daje to współrzędną z , czyli głębokość, potrzebną w obliczeniach widoczności.

Wszystkie atrybuty dodatkowe są interpolowane podobnie jak współrzędne jednorodne. Jeśli atrybut ma kwalifikator **operspective**, to parametry t i s są obliczane na podstawie kartezjańskich współrzędnych wierzchołków trójkąta — skutkuje to innym wynikiem interpolacji.

59

Obcinanie odcinka

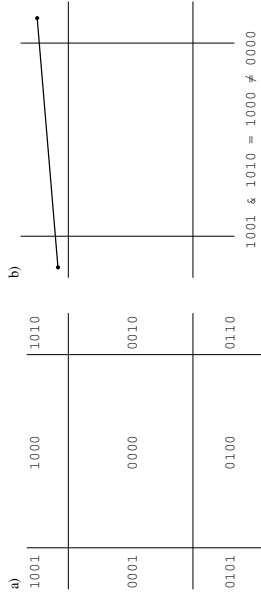
Zaczynamy od znajdowania części wspólnej odcinka na płaszczyźnie, danego przez punkty końcowe P_0, P_1 , z prostokątem (klatką w oknie).

Historycznie najstarszy jest algorytm Sutherland–Cohena, który dzieli płaszczyznę na 9 obszarów prostymi, na których leżą boki prostokąta.

Z każdą prostą jest związany 1 bit w kodzie przyporządkowanym dowolnemu punktowi płaszczyzny; są więc czterobitowe kody. Bit jest zerem, jeśli punkt leży po „właściwej” stronie prostej i jedynką w przeciwnym razie.

60

Pierwszym krokiem jest znalezienie kodów odpowiadających końcom odcinka. Jeśli koniunkcja bitowa obu kodów nie jest zerem, to oba końce odcinka leżą po „niewłaściwej” stronie pewnej prostej i wtedy cały odcinek należy odrzucić.



Jeśli oba kody są równe 0, to cały odcinek mieści się w prostokącie.

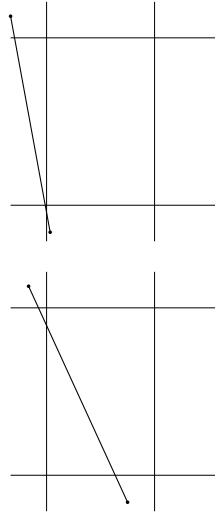
61

Jeśli któryś kod nie jest zerem, ale koniunkcja bitowa kodów jest zerem, to: jeśli kod punktu p_0 jest zerem — zamieniamy punkty i ich kody.

Następnie, na podstawie kodu, wyznaczamy prostą przecinającą odcinek i znajdujemy punkt przecięcia. Zastępujemy punkt p_0 przez ten punkt i znajdujemy jego kod. Bit odpowiadający tej prostej w tym kodzie ma wartość 0 (to wymuszamy, aby skompensować błędy zaokrągleń).

62

I powtarzamy obliczenie — aż znajdziemy przecięcie odcinka z prostokątem lub odrzucimy wszystkie jego kawałki.



Algorytm jest dość kosztowny, bo wyznacza się w nim pełne kody, których niektóre bity mogą w dalszych obliczeniach być niepotrzebne. Ponadto jest tu kumulacja błędów zaokrągleń — odrzucamy początkową reprezentację odcinka (tj. oryginalne końce) i drugi i następne punkty przecięcia wyznaczamy na podstawie punktów nieoryginalnych.

63

Algorytm Lianga–Barsky’ego

W tym algorytmie punkty p_0 i p_1 utrzymujemy aż do ostatecznego obliczenia końców przecięcia odcinka z prostokątem i wykonujemy tylko obliczenia konieczne w danym etapie. Część odcinka reprezentujemy za pomocą dwóch liczb, t_0 i t_1 , które określają przedział zmienności parametru: mamy odcinek $\{P = (1-t)p_0 + tp_1; t \in [t_0, t_1]\}$.

Początkowo $t_0 = 0$, $t_1 = 1$, później t_0 może być zwiększane a t_1 zmniejszane. Zauważmy, że możemy przyjąć dowolne początkowe wartości tych parametrów, poddając obcinaniu dowolny odcinek prostej p_0p_1 . Jeśli przyjmujemy początkowo $t_0 = -\infty$, $t_1 = \infty$, to mamy algorytm obcinania prostej.

64


```

char LBTTest ( float p, float q, float *t0, float *t1 )
{
    float r;
    if ( p < 0.0 ) {
        if ( ( r = q/p ) > *t1 ) return false;
        else if ( r > *t0 ) *t0 = r;
    }
    else if ( p > 0.0 ) {
        if ( ( r = q/p ) < *t0 ) return false;
        else if ( r < *t1 ) *t1 = r;
    }
    else return q < 0.0;
    return true;
} /*LBTTest*/

```

65

```

char LBClip ( punkt *p0, punkt *p1 )
{
    float t0, t1, dx, dy;
    t0 = 0.0; t1 = 1.0; dx = p1->x - p0->x;
    if ( LBTTest ( -dx, p0->x - xmin ) )
        if ( LBTTest ( dx, xmax - p0->x ) ) {
            dy = p1->y - p0->y;
            if ( LBTTest ( -dy, p0->y-ymin ) )
                if ( LBTTest ( dy, ymax-p0->y ) ) {
                    if ( t1 != 1.0 )
                        { p1->x = p0->x + t1*dx; p1->y = p0->y + t1*dy; }
                    if ( t0 != 0.0 )
                        { p0->x += t0*dx; p0->y += t0*dy; }
                    return true;
                }
            }
        }
    return false;
} /*LBClip*/

```

66

Odpowiednio uogólniając przedstawione wyżej algorytmy, można je dostosować do obcinania odcinków w przestrzeni (o dowolnym wymiarze) do wielościennego obszaru wypukłego. Taki obszar jest przecięciem skończenie wielu półprzestrzeni. Każdy punkt końcowy obciętego odcinka jest albo oryginalnym końcem albo punktem przecięcia odcinka z hiperpłaszczyzną będącą brzegiem jednej z półprzestrzeni.

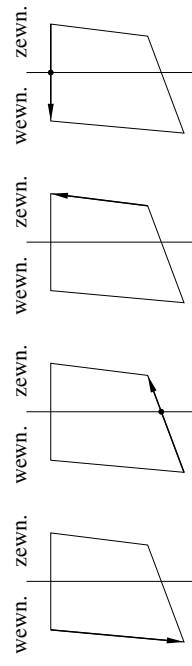
Łatwo można też tak zmienić algorytmy, aby wykonywały obliczenia we współrzędnych jednorodnych.

67

Obcinanie wielokątów

Algorytm Sutherlanda–Hodgmana ma na celu znalezienie przecięcia wielokąta z półprzestrzenią. Jeśli ma być znalezione przecięcie danego wielokąta z wielościennem wypukłym, to trzeba wykonać ten algorytm tyle razy, ile wielościan ma ścian (chyba, że wcześniej zostanie odrzucony w całości).

Rozróżniamy strony hiperpłaszczyzny, czyli półprzestrzenie, jedną z nich nazywając „wewnętrzną”.



68

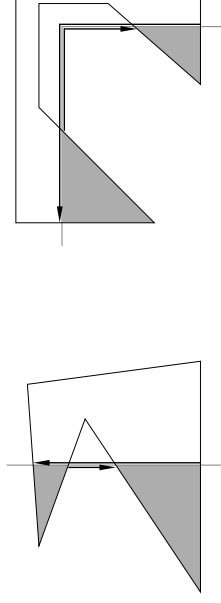
```

void SHClip ( int n, punkt w[] )
{
    s = w[n-1];    is = Inside ( s );
    for ( i = 0; i < n; i++ ) {
        p = w[i];    ip = Inside ( p );
        if ( is ) {
            if ( ip ) Output ( p );
            else Output ( q = Intersect ( s, p ) );
        }
        else if ( ip ) {
            Output ( q = Intersect ( s, p ) );
            Output ( p );
        }
        s = p;    is = ip;
    }
} /*SHClip*/

```

69

Pomocnicze procedury **Inside**, **Intersect** i **Output** odpowiednio sprawdzają, po której stronie jest dany punkt, znajdują przecięcie odcinka z hiperpłaszczyzną i wyprowadzają kolejny punkt — wierzchołek części wspólnej wielokąta z półprzestrzenia.



Jeśli dany wielokąt nie jest wypukły, to jego przecięcie z obszarem wielościennym może być niespójne. Wtedy pojawiają się „fałszywe krawędzie” obciętego wielokąta. Ten problem nie dotyczy obcinanych trójkątów.

70

Algorytm Weilera–Athertona służy do znajdowania przecięcia, a także sumy i różnicy dwóch dowolnych wielokątów. Mogą one być niespójne lub niejednospójne. W odróżnieniu od poprzedniego algorytmu, rozwiązywane zadanie jest płaskie, ponieważ istotną rolę odgrywa tu **orientacja brzegu** każdego z wielokątów, a ona jest określona w płaszczyźnie. Oczywiście, płaszczyzna ta może być zanurzona w przestrzeni trój- i więcejwymiarowej.

Zakładamy, że brzeg każdego wielokąta składa się z jednej lub kilku łamanych zamkniętych. Każda krawędź brzegu jest zorientowana (ma początek i koniec). Idąc po brzegu zgodnie z jego orientacją, wewnątrz wielokąta mamy zawsze po prawej stronie.

Z każdego wierzchołka wychodzi i do każdego wchodzi dokładnie jedna krawędź.

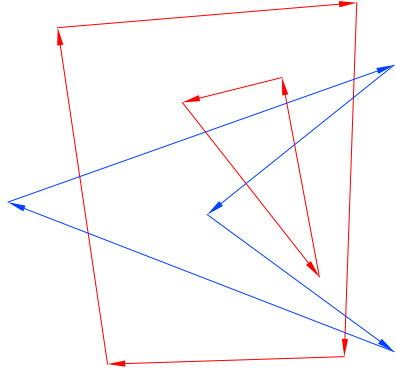
71

Dla takiej reprezentacji wielokąta łatwo jest znaleźć jego dopełnienie — wystarczy odwrócić orientację wszystkich krawędzi. Dlatego mając procedurę znajdowania przecięcia wielokątów możemy wyznaczyć także ich sumę (dopełnienie przecięcia dopełnień) i różnicę (przecięcie pierwszego wielokąta z dopełnieniem drugiego).

Algorytm buduje pewien graf skierowany, którego wierzchołkami są wierzchołki danych wielokątów i punkty przecięcia krawędzi, a krawędziami są krawędzie wielokątów lub ich fragmenty. Następnie graf ten jest przeszukiwany w celu wyprowadzenia łamanych, z których składa się brzeg przecięcia.

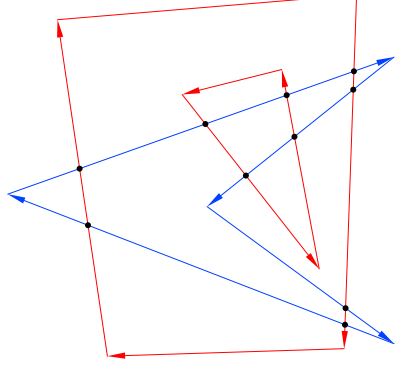
72

Początkowo graf jest sumą grafów reprezentujących brzegi wielokątów:



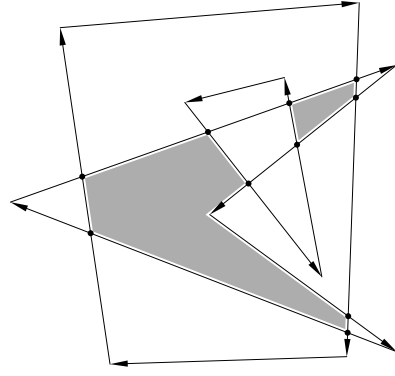
73

Nowe wierzchołki dzielą krawędzie w punktach przecięcia:



74

Graf jest przeszukiwany metodą DFS:



75

Zaznaczamy wszystkie wierzchołki jako nieodwiedzone. Zaczynając od dowolnego nieodwiedzonego wierzchołka, badamy, czy jest on wierzchołkiem przecięcia wielokątów. Jest nim każdy wierzchołek na przecięciu krawędzi, lub wierzchołek dowolnego wielokąta znajdujący się we wnętrzu drugiego.

Obchodzimy graf zgodnie z orientacją krawędzi, aż trafimy do wierzchołka, z którego wyszliśmy. Jeśli to jest wierzchołek przecięcia, to wychodzą z niego dwie krawędzie, z których jedna wchodzi do wnętrza jednego z wielokątów — i po niej idziemy.

Przechodząc przez wierzchołek przecięcia, zawsze przeskakujemy na wychodzącą z niego krawędź innego wielokąta niż ten, po którego krawędzi szliśmy. Wierzchołki zaznaczamy jako odwiedzone i wyprowadzamy.

Algorytm zatrzymuje się, gdy już nie ma nieodwiedzonych wierzchołków przecięcia.

76

Jeśli wielokąty mają odpowiednio m i n krawędzi, to koszt algorytmu może być rzędu mn i jest to koszt optymalny, jeśli każda krawędź jednego wielokąta przecina wszystkie krawędzie drugiego. Często punktów przecięcia jest mniej. Etap znajdowania przecięć krawędzi można przyspieszyć, korzystając z takich technik geometrii obliczeniowej, jak zamiatanie lub użycie drzewa binarnego podziału płaszczyzny.

Drugi najbardziej kosztowny element algorytmu to badanie, czy dany punkt (wierzchołek wielokąta) leży wewnątrz (drugiego) wielokąta. Koszt samego przeszukiwania grafu jest proporcjonalny do objętości danych opisujących wynik.

Najwięcej problemów sprawiają błędy zaokrągleń, gdy pewien wierzchołek leży na lub bardzo blisko krawędzi, której nie jest końcem. Także problematyczne bywają sytuacje, gdy pewne krawędzie wielokątów pokrywają się. Obsługa tych sytuacji komplikuje implementację tego algorytmu, ale nie można ich unikać.

77

Obcinanie w OpenGL-u jest wykonywane jako ostatni etap części przedniej potoku przetwarzania grafiki. Wszystkie punkty, odcinki i trójkąty, które trafiają do tego etapu, są obcinane do kostki standardowej. Oprócz sześciu płaszczyzn kostki, aplikacja może określić pewną liczbę dodatkowych płaszczyzn obcinających. Robi się to tak:

Równanie płaszczyzny przechodzącej przez punkt $p_0 = (x_0, y_0, z_0)$, której wektorem normalnym jest $n = (a, b, c)$ ma postać

$$ax + by + cz + d = 0,$$

gdzie $d = -ax_0 - by_0 - cz_0 = \langle n, o - p_0 \rangle$. We współrzędnych jednorodnych jest

$$aX + bY + cZ + dW = 0. \quad (*)$$

Niech $W > 0$. Zależnie od znaku wyrażenia po lewej stronie, punkt

$P = (X/W, Y/W, Z/W)$ leży na płaszczyźnie obcinającej (jeśli 0), po „właściwej” stronie (jeśli znak jest dodatni) lub po „niewłaściwej” (jeśli ujemny).

78

W treści ostatniego szadera części przedniej trzeba napisać deklarację

```
out float glClipDistance[1];
```

(lub podać inną długość tej tablicy, jeśli chcemy wprowadzić więcej płaszczyzn obcinających) i wyprowadzając wierzchołek, przypisać elementom tablicy odpowiednie wartości.

W aplikacji, przed rysowaniem, trzeba „włączyć” obcinanie; dla i -tej płaszczyzny robi to instrukcja

```
glEnable ( GL_CLIP_DISTANCE + i );
```

(wyłącza się za pomocą procedury `glDisable`).

80

Dla wierzchołka (odcinka lub trójkąta) trzeba podać wartość wyrażenia (*) lub dowolnego innego — pełni ono rolę odległości ze znakiem od płaszczyzny obcinającej. Jeśli wierzchołki odcinka lub krawędzi mają różne znaki, to na podstawie interpolacji tych odległości OpenGL wyznaczy punkt na płaszczyźnie i odrzuci odpowiednią część odcinka (lub trójkąta).

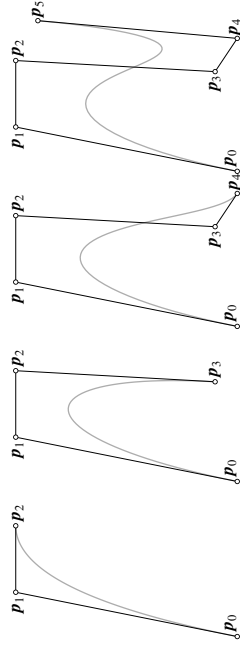
W wyniku obcinania trójkąta do kostki standardowej może powstać nawet osmitokąt wypukły, a każda dodatkowa płaszczyzna obcinająca może sprawić, że powstanie wielokąt wypukły o jeszcze jednym wierzchołku. Taki wielokąt jest dzielony na trójkąty, przekazywane następnie do etapu rasteryzacji.

Podczas obcinania następuje interpolacja wszelkich atrybutów wierzchołków — również tu korzysta się z odległości ze znakiem (*).

79

Elementy modelowania geometrycznego

Krzywa Béziera stopnia n jest to reprezentacja parametrycznej krzywej wielomianowej w postaci ciągu $n + 1$ punktów kontrolnych. Tak zwana łamana kontrolna, której wierzchołkami są te punkty, przybliża kształt krzywej.



81

Parametryzacja krzywej Béziera jest dana wzorem

$$\mathbf{p}(t) = \sum_{i=0}^n \mathbf{p}_i B_i^n(t),$$

w którym występują wielomiany bazowe Bernsteina

$$B_i^n(t) \stackrel{\text{def}}{=} \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \dots, n.$$

Zatem istotnie, jest to parametryzacja wielomianowa stopnia co najwyżej n .

Dziedziną parametryzacji może być dowolny przedział $[a, b]$, ale zazwyczaj przyjmuje się, że $t \in [0, 1]$.

82

Aby zbadać własności reprezentacji i otrzymać z niej użyteczne algorytmy, udowodnimy wzór rekurencyjny

$$B_0^0(t) = 1, \\ B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) \quad \text{dla } n > 0,$$

w którym przyjmujemy umowę, że $B_i^n(t) \equiv 0$ dla $i < 0$ oraz $i > n$.

Bezpośrednio sprawdzamy, że $B_0^0(t) = \binom{0}{0} t^0 (1-t)^0$. Dla $n > 0$ oraz $i \in \{1, \dots, n-1\}$ piszemy

$$(1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) = \\ (1-t) \binom{n-1}{i} t^i (1-t)^{n-i-1} + t \binom{n-1}{i-1} t^{i-1} (1-t)^{n-i} = \\ \left(\binom{n-1}{i} + \binom{n-1}{i-1} \right) t^i (1-t)^{n-i} = \binom{n}{i} t^i (1-t)^{n-i} = B_i^n(t).$$

Jeśli $i = 0$ lub $i = n$, to ten rachunek możemy wykonać, pomijając składnik, w którym występuje czynnik zerowy $B_{n-1}^{n-1}(t)$ lub $B_0^{n-1}(t)$. Jest $\binom{n-1}{0} = \binom{n-1}{n-1} = \binom{n}{0} = \binom{n}{n} = 1$. \square

83

Mając udowodniony wzór, dla $n > 0$ obliczmy

$$\begin{aligned} \mathbf{p}(t) &= \sum_{i=0}^n \mathbf{p}_i B_i^n(t) = \mathbf{p}_0 B_0^n(t) + \sum_{i=1}^{n-1} \mathbf{p}_i B_i^n(t) + \mathbf{p}_n B_n^n(t) \\ &= (1-t) \mathbf{p}_0 B_0^{n-1}(t) + \sum_{i=1}^{n-1} \mathbf{p}_i \left((1-t) B_i^{n-1}(t) + t B_{i-1}^{n-1}(t) \right) + t \mathbf{p}_n B_{n-1}^{n-1}(t) \\ &= (1-t) \sum_{i=0}^{n-1} \mathbf{p}_i B_i^{n-1}(t) + t \sum_{i=0}^{n-1} \mathbf{p}_{i+1} B_i^{n-1}(t). \end{aligned}$$

Punkt $\mathbf{p}(t)$ możemy zatem otrzymać, dokonując interpolacji między punktami dwóch krzywych Béziera stopnia $n-1$: pierwsza z nich jest reprezentowana przez punkty $\mathbf{p}_0, \dots, \mathbf{p}_{n-1}$, a druga przez punkty $\mathbf{p}_1, \dots, \mathbf{p}_n$.

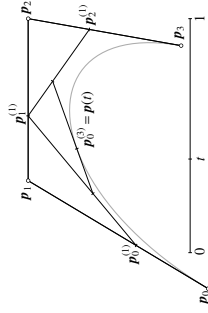
Idąc dalej tym tropem, możemy dojść do punktów na krzywych stopnia 0: dla każdego i jest $\sum_{i=0}^0 \mathbf{p}_i B_0^0(t) = \mathbf{p}_i$.

84

Stąd wynika algorytm de Casteljau: mając dane punkty p_0, \dots, p_n i liczbę t , oznaczmy $p_i^{(0)} = p_i$ i obliczmy

$$\begin{aligned} \text{for } (j = 1; j \leq n; j++) \\ \text{for } (i = 0; i \leq n-j; i++) \\ p_i^{(j)} = (1-t)p_i^{(j-1)} + tp_{i+1}^{(j-1)}; \end{aligned}$$

Ostatni obliczony punkt to $p_0^{(n)} = p(t)$.



85

Własności krzywych Béziera:

- Dla każdego n i t jest $\sum_{i=0}^n B_i^n(t) = 1$, stąd krzywa i jej punkty kontrolne znajdują się w tej samej przestrzeni. Co więcej, jeśli f jest dowolnym przekształceniem afinicznym, to obraz w tym przekształceniu krzywej p , reprezentowanej przez punkty p_0, \dots, p_n , jest reprezentowany przez punkty $f(p_0), \dots, f(p_n)$. Reprezentacja Béziera krzywej jest **afinicznie niezmiennicza**.
- Jeśli $t \in [0, 1]$, dla $i \in \{0, \dots, n\}$ jest $B_i^n(t) \geq 0$. Razem z poprzednią własnością oznacza to, że wszystkie punkty $p(t)$ krzywej, dla $t \in [0, 1]$, leżą w **otoczce** wypukłej zbioru punktów kontrolnych.

86

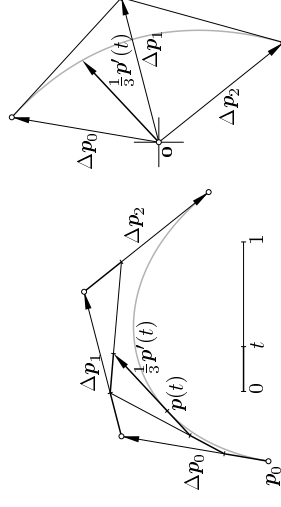
- Jest $B_0^n(0) = 1, B_1^n(0) = \dots = B_{n-1}^n(0) = 0$ oraz $B_0^n(1) = \dots = B_{n-1}^n(1) = 0, B_n^n(1) = 1$. Stąd wynika interpolacja skrajnych punktów kontrolnych: $p(0) = p_0, p(1) = p_n$.

- Łatwo można udowodnić, że $\frac{d}{dt} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$ (zostawiam to jako ćwiczenie, uwaga: $B_{-1}^{n-1}(t) = B_n^{n-1}(t) = 0$). Można stąd obliczyć

$$\begin{aligned} p'(t) &= \sum_{i=0}^n p_i (B_i^n(t))' \\ &= n(-p_0 B_0^{n-1}(t) + \sum_{i=1}^{n-1} p_i (B_{i-1}^{n-1}(t) - B_i^{n-1}(t)) + p_n B_{n-1}^{n-1}(t)) \\ &= \sum_{i=0}^{n-1} n(p_{i+1} - p_i) B_i^{n-1}(t). \end{aligned}$$

Zatem pochodna krzywej stopnia n jest krzywą stopnia $n-1$, której punktami są wektory $n(p_1 - p_0), \dots, n(p_n - p_{n-1})$.

87



- Możemy też zobaczyć, że

$$p'(t) = n \left(\sum_{i=0}^n p_{i+1} B_i^{n-1}(t) - \sum_{i=0}^{n-1} p_i B_i^{n-1}(t) \right) = n(p_{i+1} - p_0).$$

Wektor pochodnej dla danego t otrzymamy, mnożąc przez n różnicę punktów otrzymanych w przedostatnim kroku algorytmu de Casteljau.

88

- Algorytm de Casteljau umożliwia **podział krzywej Béziera**. Jeśli $0 < t < 1$, to otrzymane przez ten algorytm punkty $P_0^{(0)}, \dots, P_n^{(0)}$ oraz $P_0^{(n)}, \dots, P_n^{(n)}$ reprezentują łuki krzywej nad przedziałami $[0, t]$ i $[t, 1]$. Można wprowadzić lokalne parametry, np. s i u zmieniające się w tych przedziałach od 0 do 1.

Aby podzielić krzywą, można wykonać procedurę

```
void Podziel1 ( int n, punkt p[], float t )
{
    q[0] = p[0];
    for ( j = 1; j <= n; j++ ) {
        for ( i = 0; i <= n-j; i++ )
            p[i] = (1-t)*p[i] + t*p[i+1];
        q[j] = p[0];
    }
} /*Podziel1*/
```

Do tablicy q trafiają punkty reprezentujące pierwszą część łuku, a punkty dane w tablicy p są zastępowane przez punkty kontrolne drugiego łuku.

89

- Podaną wyżej procedurę można zastosować do **rekurencyjnego podziału krzywej**, w celu jej narysowania jako łamanej złożonej z dostatecznie krótkich odcinków — podobnie jak we wcześniejszym podanym algorytmie rysowania elips.

Kryterium zatrzymania rekurencji może być takie: jeśli odległość punktów kontrolnych P_1, \dots, P_{n-1} od odcinka $\overline{P_0P_n}$ jest mniejsza niż ε , to (z własności otoczki wypukłej) wszystkie punkty łuku są w odległości mniejszej niż ε od tego odcinka. Można przyjąć ε bliski średnicy jednego piksela.

- Z żadną prostą (na płaszczyźnie) ani z żadną płaszczyzną (w przestrzeni trójwymiarowej) krzywa nie ma więcej punktów wspólnych niż jej łamana kontrolna. To jest tzw. **własność zmniejszania wariacji**.

90

- Wielomiany bazowe Bernsteina spełniają też formułę

$$B_i^n(t) = \frac{n-i}{n+1-i} B_i^{n+1}(t) + \frac{i-1}{n+1} B_{i-1}^{n+1}(t).$$

Podstawiając ją do wzoru definiującego krzywą, po przekształceniach otrzymamy

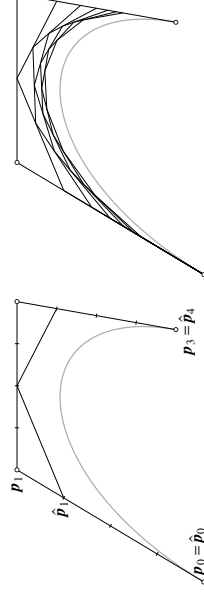
$$p(t) = \sum_{i=0}^n p_i B_i^n(t) = \sum_{i=0}^{n+1} \hat{p}_i B_i^{n+1}(t).$$

gdzie

$$\hat{p}_i = \frac{n+1-i}{n+1} p_i + \frac{i}{n+1} p_{i-1}.$$

W ten sposób można dokonać **podwyższenia stopnia** krzywej, tj. przejścia do reprezentacji w bazie wyższego stopnia.

91



- Ze wzoru na podwyższenie stopnia wynika, że jeśli punkty P_0, \dots, P_n są współliniowe, uporządkowane na prostej i równoodległe, to krzywa Béziera jest odcinkiem sparametryzowanym ze stałą prędkością.

92

- Koszt algorytmu de Casteljau jest rzędu n^2 , ale oprócz punktu $p(t)$ daje on wiele innych wyników (podział krzywej, pochodne). Algorytm o koszcie rzędu n , który dostarcza tylko punkt krzywej, opiera się na schemacie Hornera. Niech $s = 1 - t$. Wtedy

$$p(t) = p_0 \binom{n}{0} s^n + p_1 \binom{n}{1} t s^{n-1} + \dots + p_{n-1} \binom{n}{n-1} t^{n-1} s + p_n \binom{n}{n} t^n =$$

$$(\dots (p_0 \binom{n}{0} s + p_1 \binom{n}{1} t) s + \dots + p_{n-1} \binom{n}{n-1} t^{n-1}) s + p_n \binom{n}{n} t^n.$$

Mozemy użyć wzorów $\binom{n}{0} = 1, \binom{n}{i+1} = n$ oraz $\binom{n}{i+1} = \frac{n-1}{i+1} \binom{n}{i}$ i dostać algorytm

```

s = 1-t;  p = p0;  d = t;  b = n;
for ( i = 1; i <= n; i++ ) {
    p = s*p + b*d*p_i;
    d *= t;  b = (b*(n-1))/(i+1);
}

```

Krzywe B-sklejane

Krzywe Béziera można łączyć, co ułatwia modelowanie obiektów o skomplikowanych kształtach: z interpolacji skrajnych punktów kontrolnych wynika łatwość „sklejania” krzywych w sposób ciągły, a z faktu, że krzywa jest styczna w tych punktach do skrajnych odcinków lamanej kontrolnej wynika łatwość uzyskania krzywej gładkiej (tj. której styczna ma kierunek zmieniający się w sposób ciągły). Metodę osiągnięcia gładkości połączenia można uogólnić na pochodne wyższego rzędu, ale to przestaje być wygodne.

Bardziej systematyczny sposób otrzymywania gładkich krzywych polega na użyciu **reprezentacji B-sklejanej**. Krzywa taka składa się z łuków wielomianowych stopnia n ; liczba n może być niezbyt duża, ale łuków może być dowolnie wiele.

Aby określić krzywą B-sklejaną, należy podać **stopień n** , niemający ciąg węzłów u_0, \dots, u_N oraz **ciąg punktów kontrolnych d_0, \dots, d_{N-n-1}** . Parametryzacja krzywej jest dana wzorem

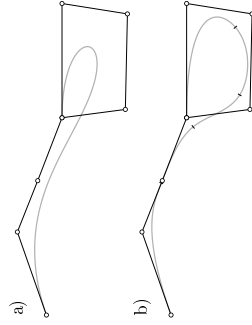
$$s(t) = \sum_{i=0}^{N-n-1} d_i N_i^n(t), \quad t \in [u_n, u_{N-n}),$$

w którym występują **unormowane funkcje B-sklejane stopnia n** , określone przez liczbę n i ciąg węzłów; wzorem **Mansfielda-de Boora-Coxa**:

$$N_i^0(t) = \begin{cases} 1 & \text{dla } t \in [u_i, u_{i+1}), \\ 0 & \text{w przeciwnym razie,} \end{cases}$$

$$N_i^n(t) = \frac{t - u_i}{u_{i+n} - u_i} N_i^{n-1}(t) + \frac{u_{i+n+1} - t}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(t) \quad \text{dla } n > 0.$$

Wzór ten jest uogólnieniem wzoru rekurencyjnego udowodnionego wcześniej dla wielomianów bazowych Bernsteina.



Jeśli kształt ma być skomplikowany, to trzeba dużo punktów kontrolnych. Dla krzywej Béziera to wymusza wysoki stopień, a wtedy kształt krzywej może mieć niewiele wspólnego z kształtem lamanej. Lepiej to wygląda dla krzywej B-sklejanej.

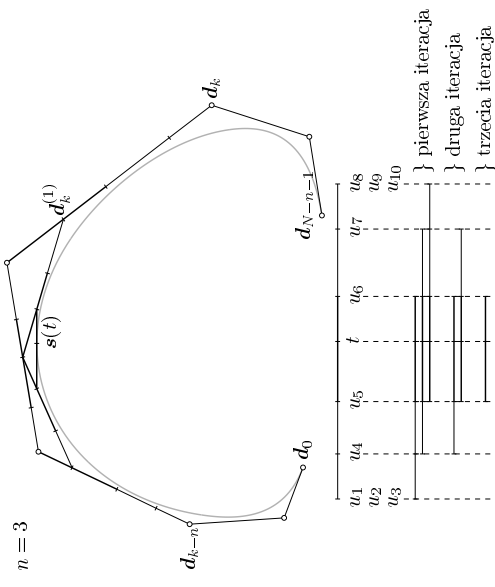
W podobny sposób, jak dla krzywych Béziera, na podstawie wzoru Mansfielda-de Boora-Coxa wyprowadza się algorytm de Boora, który dla danej krzywej s i liczby $t \in [u_n, u_{N-1}]$ oblicza punkt $s(t)$:

```

/* znajdź takie  $k$ , że  $t \in [u_k, u_{k+1})$  */
/* oznacz  $d_i^{(0)} = d_i$  dla  $i = k - n, \dots, k$  */
for ( r = 0; r < N && t == u_{k-r}; r++ );
for ( j = 1; j < n-r; j++ )
    for ( i = k-n+j; i <= k-r; i++ ) {
         $\alpha_i^{(j)} = (t - u_i) / (u_{i+n+1-j} - u_i)$ ;
         $d_i^{(j)} = (1 - \alpha_i^{(j)}) d_{i-1}^{(j-1)} + \alpha_i^{(j)} d_i^{(j-1)}$ ;
    }
/*  $s(t) = d_{k-r}^{(n-r)}$  */

```

97

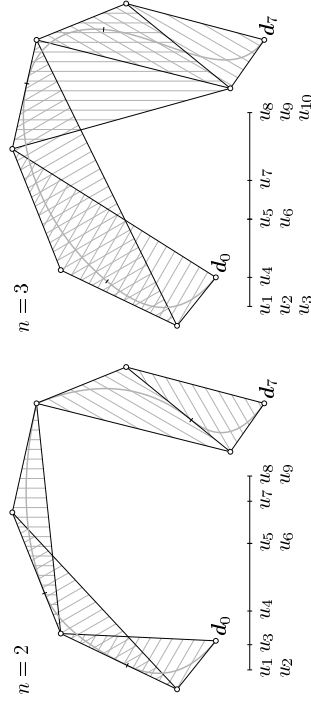


98

Własności krzywych B-sklejanych:

- Jeśli wszystkie węzły od u_n do u_{N-n} są różne, to krzywa składa się $N - 2n$ łuków wielomianowych; w przeciwnym razie (gdy występują węzły krotne) liczba łuków jest mniejsza.
- Algorytm de Boora dokonuje interpolacji kolejno otrzymywanych punktów, przy czym $\alpha_i^{(j)} \in [0, 1]$, zatem wszystkie otrzymane punkty leżą w otocze wypukłej punktów d_{k-n}, \dots, d_k . Cały łuk dla $t \in [u_k, u_{k+1})$ jest położony w otocze wypukłej tych punktów kontrolnych — to jest silna własność otoczki wypukłej.

99



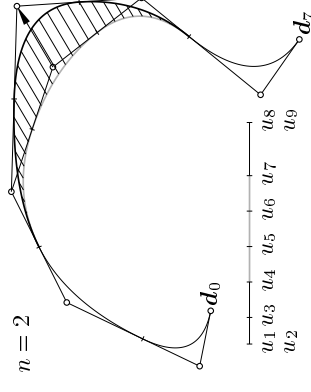
100

- Suma funkcji B-sklejanych w przedziale $[u_k, u_{k+1})$ jest stała, równa 1. Mamy stąd **afiniczną niezmienniczość** reprezentacji: aby poddać krzywą dowolnemu przekształceniu afinicznemu, wystarczy zastosować to przekształcenie do jej punktów kontrolnych.

- Reprezentacja zapewnia **lokalną kontrolę kształtu**: ponieważ punkt $s(t)$ dla $t \in [u_k, u_{k+1})$ zależy tylko od punktów kontrolnych d_{k-n}, \dots, d_k , przemieszczenie każdego innego punktu nie zmienia łuku krzywej dla $t \in [u_k, u_{k+1})$.

W konsekwencji, przemieszczenie punktu kontrolnego d_i zmienia tylko łuk krzywej odpowiadający wartościom parametru $t \in [u_n, u_{N-n}) \cap [u_i, u_{i+n+1})$.

101



102

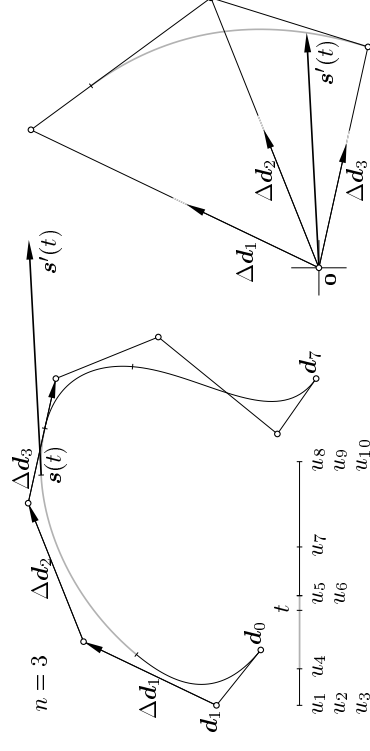
- Krzywa składa się z łuków wielomianowych stopnia co najwyżej n , a zatem pochodna parametryzacji jest krzywą sklejaną stopnia mniejszego niż n . Da się ją przedstawić w postaci

$$s'(t) = \sum_{i=0}^{N-n-2} \frac{n}{u_{i+n+1} - u_{i+1}} (d_{i+1} - d_i) N_{i+1}^{n-1}(t).$$

Ciąg węzłów użyty do określenia funkcji N_i^{n-1} jest ten sam, co ciąg określający funkcje $N_i^n(t)$.

Zastosowanie silnej własności otoczki wypukłej do pochodnej daje **silną własność hodografu** krzywych B-sklejanych: dla $t \in (u_k, u_{k+1})$ wektor $s'(t)$ jest kombinacją liniową wektorów różnic $d_{k-n+1} - d_{k-n}, \dots, d_k - d_{k-1}$ o dodatnich współczynnikach.

103



104

- W otoczeniu węzła o krotności r parametryzacja jest klasy C^{n-r} . Jeśli $r = n$, to parametryzacja jest ciągła, ale jej pochodna nie musi być ciągła. Dla $r = n - 1$ pochodna jest już ciągła. Dla $r = n - 2$ mamy też gwarancję ciągłości pochodnej drugiego rzędu. W szczególności, krzywa B-sklejana stopnia 3 (kubiczna) o węzłach jednokrotnych jest klasy C^2 ; jeśli jej pochodna nie znika, to taka krzywa ma ciągłą krzywiznę.

- Jeśli dwa sąsiednie węzły mają krotność co najmniej n , tj.

$u_{k-n+1} = \dots = u_k < u_{k+1} = \dots = u_{k+n}$, to lamana kontrolna krzywej zawiera reprezentację Béziera łuku wielomianowego dla $t \in [u_k, u_{k+1}]$. Mamy wtedy

$$s(t) = \sum_{i=0}^n d_{k-n+i} B_i^n(v), \quad \text{gdzie } v = \frac{t - u_k}{u_{k+1} - u_k}.$$

- Jeśli pewien węzeł ma krotność n , to odpowiedni punkt kontrolny jest również punktem krzywej. Dokładniej, jeśli $u_{k-n+1} = \dots = u_k < u_{k+1}$, to $s(u_k) = d_{k-n}$.

105

Wstawianie węzła do krzywej B-sklejanej (W. Boehm, 1980) jest metodą otrzymania reprezentacji *tej samej* krzywej (o niezminionej parametryzacji) w nowej bazie — dołączenie węzła powoduje powiększenie o 1 wymiaru przestrzeni funkcji sklejaných. Zakładamy, że nowym węzłem jest liczba $t \in [u_n, u_{N-n}]$.

1. Określamy tzw. **współrzędne Greville'a**:

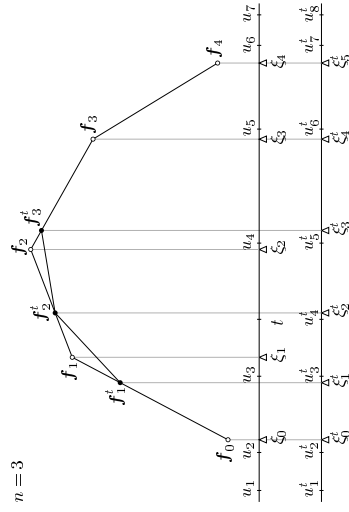
$$\xi_i = \frac{1}{n}(u_{i+1} + \dots + u_{i+n}).$$

Będziemy dokonywać przekształceń lamanej o wierzchołkach $f_i = (\xi_i, d_i)$, $i = 0, \dots, N - n - 1$.

2. Liczbę $t \in [u_n, u_{N-n}]$ dołączamy do początkowego ciągu węzłów, z zachowaniem uporządkowania.

106

3. Obliczamy współrzędne Greville'a ξ_i^t dla nowego ciągu węzłów.
4. Na lamanej znajdujemy punkty $f_i^t = (\xi_i^t, d_i^t)$. Punkty d_i^t są punktami kontrolnymi nowej reprezentacji krzywej.



107

Implementacja nie wymaga jawnego obliczania współrzędnych Greville'a. Nowa reprezentacja jest wpisywana na miejscu starej:

```

/* u[i] = u_i dla i = 1, ..., N-1, d[i] = d_i dla i = 0, ..., N-n-1, */
/* t \in [u_n, u_{N-n}] */
k = N - 1;
while ( t < u[k] ) k --;
r = 0; i = k;
while ( i \ge 1 && t = u[i] ) { i --; r ++; }
for ( i = N-n-1; i \ge k-r; i -- ) d[i+1] = d[i];
for ( i = k-r; i \ge k-n+1; i -- )
    d[i] = ((u[i+n] - t) * d[i-1] + (t - u[i]) * d[i]) / (u[i+n] - u[i]);
for ( i = N-1; i \ge k+1; i -- ) u[i+1] = u[i];
u[k+1] = t;
N ++;
/* zmienna N oraz tablice u i d zawierają wynik. */

```

108

Własności wstawiania węzła:

- Liczby węzłów i punktów kontrolnych są zwiększane o 1.
- Wynik wstawienia wielu węzłów nie zależy od kolejności ich wstawiania.
- Parametryzacja pozostaje niezmienniona (zmienia się tylko jej reprezentacja).
- Algorytm de Boora obliczania punktu $s(t)$ jest równoważny wstawianiu węzła — liczby t — tyle razy, aby węzeł t miał krotność n .
- Wstawiając węzły, które tworzą zbiór gęsty w przedziale $[u_n, u_{N-n})$, otrzymujemy ciąg lamanych kontrolnych zbiegający do krzywej. Odległość lamanej od krzywej ma oszacowanie proporcjonalne do h_{\max}^2 , gdzie h_{\max} jest maksymalną długością przedziału między węzłami.

109

Zastosowania:

- Mając reprezentację z niewieloma węzłami, kształtujemy krzywą zgrubnie, a potem wstawiamy węzły i możemy cyzelować szczegóły.
- Wstawiamy dużo węzłów i rysujemy lamana zamiast krzywej, otrzymując całkiem dokładny obraz.
- Wstawiamy węzły tak, aby wszystkie miały krotność $n + 1$. Wtedy lamana kontrolna składa się z połączonych lamanych kontrolnych łuków wielomianowych w reprezentacji Béziera. Mając je, możemy szybko narysować krzywą.

110

- Są różne konstrukcje wymagające działań algebraicznych na funkcjach sklejanych (np. podwyższenie stopnia, dodawanie, mnożenie). Można wstawić węzły, wykonać działania na wielomianach (w reprezentacji Béziera), a potem usunąć nadmiarowe węzły (przez rozwiązanie pewnych układów równań liniowych).

111

Algorytm Lanéa–Riesenfelda jest metodą wstawiania wielu węzłów jednocześnie — ale krzywa B-sklejana ma być oparta na ciągu **węzłów równoodległych**. Ciąg węzłów zostaje dwukrotnie zagęszczony — nowe węzły dzielią przedziały między dotychczasowymi na połowy, zatem nowa reprezentacja krzywej też jest oparta na ciągu węzłów równoodległych. To jest sposób otrzymania ciągu lamanych szybko zbieżnego do krzywej, aby ją narysować.

Bez straty ogólności możemy przyjąć, że oryginalne węzły są kolejnymi liczbami całkowitymi, a nowe węzły będą miały część ułamkową $\frac{1}{2}$. Możemy też napisać wzór

$$s(t) = \sum_{i \in \mathbb{Z}} c_i N_i^n(t) = \sum_{i \in \mathbb{Z}} d_i M_i^n(t),$$

w którym funkcje N_i^n są oparte na węzłach oryginalnych, a funkcje M_i^n są oparte na nowym ciągu węzłów, dwukrotnie gęstszym. Nieskończenie wiele niepotrzebnych składników obu sum odrzucimy później, zostawiając tylko te, które nie znikają w przedziale $[n, N - n)$.

112

Algorytm składa się z kroku **podwajania**, po którym następuje n kroków **uśredniania**.

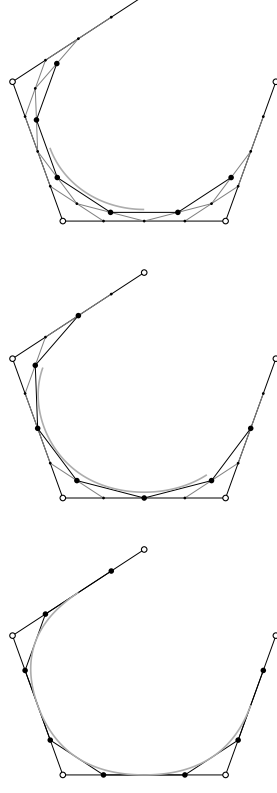
Podwajanie polega na przyjęciu dla każdego punktu c_i dwóch punktów,

$$\mathbf{d}_{2i}^{(0)} = \mathbf{d}_{2i+1}^{(0)} = \mathbf{c}_i.$$

Uśrednianie: w j -tym kroku dla każdego i obliczamy $\mathbf{d}_i^{(j)} = \frac{1}{2}(\mathbf{d}_{i-1}^{(j-1)} + \mathbf{d}_i^{(j-1)})$.

Na końcu otrzymujemy punkty $\mathbf{d}_i = \mathbf{d}_i^{(n)}$.

113



114

Wektorowe współczynniki tensorowych płatów parametrycznych, jeśli suma wszystkich elementów bazy jest równa 1, są **punktami kontrolnymi**, które przedstawiamy jako **siatkę kontrolną** płata.

W ten sposób definiujemy **płaty Béziera** stopnia (n, m) :

$$\mathbf{p}(u, v) = \sum_{i=0}^n \sum_{j=0}^m \mathbf{p}_{ij} B_i^n(u) B_j^m(v), \quad (u, v) \in [0, 1]^2$$

oraz **płaty B-sklejane** stopnia (n, m) :

$$\mathbf{s}(u, v) = \sum_{i=0}^{N-n-1} \sum_{j=0}^{M-m-1} \mathbf{d}_{ij} N_i^n(u) N_j^m(v), \quad (u, v) \in [u_n, n_{N-n}] \times [v_m, v_{M-m}].$$

Funkcje B-sklejane są określone przez ciągi węzłów u_0, \dots, u_N i v_0, \dots, v_M , w ogólności różne.

116

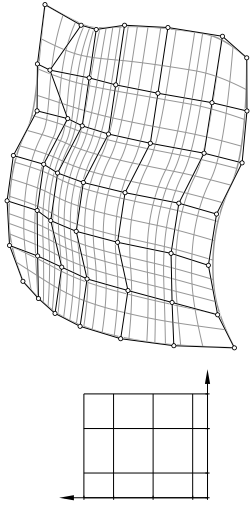
Tensorowe płaty Béziera i B-sklejane

Płat tensorowy jest parametryzacjaą otrzymaną przy użyciu bazy tensorowej — jeśli mamy dwie bazy funkcji jednej zmiennej, $\{f_0, \dots, f_n\}$ i $\{g_0, \dots, g_m\}$, to baza tensorowa składa się z wszystkich iloczynów $f_i \otimes g_j$, czyli funkcji

$$(f_i \otimes g_j)(u, v) = f_i(u)g_j(v).$$

Dziedzina tych iloczynów jest iloczynem kartezyjskim dziedzin funkcji f_i i g_j .

115



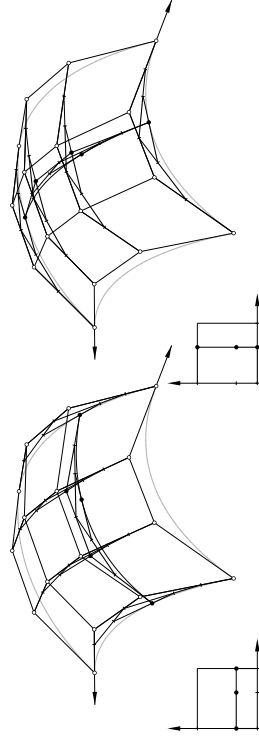
Jeśli krzywa jest porozciągany i powyginany odcinkiem, to płat tensorowy jest porozciągany i powyginany prostokątem.

Do płatów tensorowych możemy stosować wszystkie algorytmy przetwarzania krzywych. Na przykład obliczenie punktu $\mathbf{p}(u, v)$ płata Béziera dla danych liczb u, v możemy sprowadzić do znalezienia pewnej liczby punktów na krzywych Béziera:

$$\mathbf{p}(u, v) = \sum_{i=0}^n \underbrace{\left(\sum_{j=0}^m \mathbf{p}_{ij} B_j^m(v) \right)}_{\mathbf{q}_i} B_i^n(u) = \sum_{i=0}^n \mathbf{q}_i B_i^n(u).$$

Do znalezienia punktów \mathbf{q}_i możemy użyć zarówno algorytmu de Casteljau, jak i schematu Hornera. Ten sam punkt możemy też znaleźć na podstawie wzoru

$$\mathbf{p}(u, v) = \sum_{i=0}^n \underbrace{\left(\sum_{j=0}^m \mathbf{p}_{ij} B_i^n(u) \right)}_{\mathbf{r}_j} B_j^m(v) = \sum_{j=0}^m \mathbf{r}_j B_j^m(v).$$



Pochodne cząstkowe płata Béziera są opisane wzorami

$$\frac{\partial \mathbf{p}}{\partial u}(u, v) = \sum_{i=0}^{n-1} \sum_{j=0}^m n(\mathbf{p}_{i+1,j} - \mathbf{p}_{ij}) B_i^{n-1}(u) B_j^m(v),$$

$$\frac{\partial \mathbf{p}}{\partial v}(u, v) = \sum_{i=0}^n \sum_{j=0}^{m-1} m(\mathbf{p}_{i,j+1} - \mathbf{p}_{ij}) B_i^n(u) B_j^{m-1}(v),$$

a zatem możemy je obliczyć, jeśli umiemy znaleźć pochodną krzywej Béziera $\mathbf{p}(t) = \sum_{i=0}^n \mathbf{p}_i B_i^n(t)$:

$$\mathbf{p}'(t) = n \left(\sum_{i=0}^{n-1} \mathbf{p}_{i+1} B_i^{n-1}(t) - \sum_{i=0}^n \mathbf{p}_i B_i^{n-1}(t) \right) = n(\mathbf{p}_1^{(n-1)} - \mathbf{p}_0^{(n-1)}).$$

Punkty $\mathbf{p}_0^{(n-1)}$ i $\mathbf{p}_1^{(n-1)}$ są otrzymane w przedostatnim kroku algorytmu de Casteljau.

Dla płatów tensorowych Béziera możemy użyć algorytmu de Casteljau do podziału płata na kawałki (wzdłuż krzywej stałego parametru u albo v), możemy też dokonać podwyższenia stopnia.

Podobnie wszystkie algorytmy przetwarzania krzywych B-sklejanych mają zastosowanie do tensorowych płatów B-sklejanych: **znajdowanie punktu**, **obliczanie pochodnej**, **wstawianie węzła**.

Konsekwencją tensorowej definicji płatów są też ich własności przeniesione z krzywych: **afiniczna niezmienniczość reprezentacji**, **(silna) własność otoczki wypukłej** i **(silna) własność hodografu**.

121

Wymierne krzywe i płaty Béziera i B-sklejane

Krzywe i płaty wymierne otrzymamy, reprezentując punkty kontrolne za pomocą współrzędnych jednorodnych — określają one wielomianowe **krzywe i płaty jednorodne**, których punkty są wektorami współrzędnych jednorodnych odpowiednich punktów krzywych i płatów wymiernych. Na przykład dla wymiernych krzywych Béziera w przestrzeni trójwymiarowej mamy krzywą jednorodną

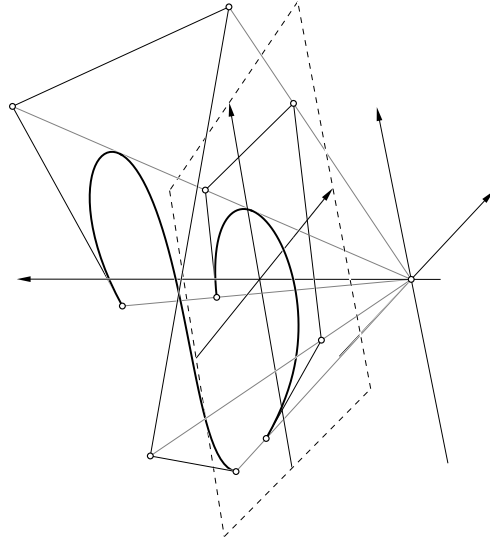
$$\mathbf{P}(t) = \sum_{i=0}^n \mathbf{P}_i B_i^n(t),$$

gdzie $\mathbf{P}_i = (X_i, Y_i, Z_i, W_i)$. Punkt $\mathbf{P}(t)$ reprezentuje punkt

$$\mathbf{P}(t) = \frac{\sum_{i=0}^n w_i \mathbf{P}_i B_i^n(t)}{\sum_{i=0}^n w_i B_i^n(t)},$$

przy czym $w_i = W_i$ oraz $\mathbf{P}_i = (x_i, y_i, z_i) = (X_i/W_i, Y_i/W_i, Z_i/W_i)$ dla $i = 0, \dots, n$ (jeśli wszystkie wagi są niezerowe).

122



123

Modelując wymierną krzywą lub tensorowy płat wymierny, rozmieszczamy w przestrzeni punkty kontrolne \mathbf{P}_i (lub $\mathbf{d}_{i,j}$), tj. ustalamy ich współrzędne kartezjańskie oraz dobieramy współrzędne wagowe.

Wszystkie algorytmy dotychczas podane możemy stosować do krzywych i płatów jednorodnych, przechodząc do współrzędnych kartezjańskich dopiero gdy trzeba krzywą lub płat narysować. Nawet tego nie trzeba robić w OpenGL-u, ponieważ do etapu obcinania wprowadza się współrzędne jednorodne wierzchołków — czyli trzeba wprowadzić odpowiednie punkty krzywych lub płatów jednorodnych.

Jeśli wszystkie wagi są jednakowe (np. równe 1), to otrzymane krzywe lub płaty są identyczne z krzywymi lub płacami wielomianowymi (lub sklejanymi) — bo wtedy mianownik w odpowiednim wzorze jest funkcją stałą.

124

Jedynie krzywe stożkowe o parametryzacjach wielomianowych to parabole.

Klasa krzywych wymiernych zawiera m.in. parametryzacje wszystkich krzywych stożkowych (okręgów, elips, hiperbol). Klasa płatów wymiernych zawiera w szczególności kwadryki (powierzchnie drugiego stopnia), a także płaty powierzchni obrotowych, których tworzące są krzywymi wymiernymi.

Dodatkową zaletą tej klasy jest **niezmienniczość rzutowa** reprezentacji. Zauważmy, że rzut perspektywny paraboli może być dowolną krzywą stożkową. Obraz w dowolnym przekształceniu rzutowym (lub w rzucie perspektywicznym) krzywej lub płata wymiennego ma parametryzację wymierną i to tego samego stopnia.

125

Zauważmy, że np. dla wymiernej krzywej Béziera możemy napisać

$$P(t) = \sum_{i=0}^n P_i \frac{w_i B_i^n(t)}{\sum_{j=0}^n w_j B_j^n(t)}.$$

Suma funkcji wymiernych, przez które mnożymy punkty P_i , jest równa 1, a zatem taka reprezentacja jest **afinicznie niezmiennicza**. Jeśli wszystkie wagi są dodatnie, to na odcinku $[0, 1]$ funkcje są nieujemne, skąd wynika **własność otoczki wypukłej** wymiernych krzywych Béziera.

Wymierne krzywe i płaty B-sklejane są często określane skrótem NURBS — to od zwrotu *nonuniform rational B-splines*, co podkreśla użycie węzłów nierównoodległych w ich reprezentacji.

126

Metody siatek

Siatka kontrolna płata B-sklejanego jest przybliżeniem tego płata i wstawianie węzłów do obu ciągów węzłów produkuje ciąg siatek zbiegający do tego płata. Wybraną siatkę z tego ciągu można „przerobić” na trójkąty i je narysować.

Sama metoda wstawiania węzłów nie jest istotna; co więcej, można określić wiele innych metod zagęszczania siatek, wytwarzających ciągi siatek zbieżne do jakiegś powierzchni. Taka powierzchnia jest *zdefiniowana* przez algorytm zagęszczania siatki.

127

Przypuśćmy, że mamy płat powierzchni B-sklejanej stopnia (n, n) , którego reprezentacja jest oparta na węzłach *równoodległych*. Każdy z tych ciągów możemy dwukrotnie zagęścić, wstawiając nowe węzły za pomocą algorytmu Boehma lub za pomocą algorytmu Lanèa–Riesenfelda. Zależnie od tego, który ciąg zagęszczamy, odpowiednio działania wykonujemy na wszystkich kolumnach albo na wszystkich wierszach siatki kontrolnej.

Algorytm Lanèa–Riesenfelda możemy zmodyfikować tak, aby wykonywać działania na wierszach i kolumnach siatki jednocześnie. Jest tak dlatego, bo każda z operacji: podwajanie i uśrednianie wierszy, jest przemienna z podwajaniem i uśrednianiem kolumn. Ponieważ algorytm dla krzywych składa się z kroku podwajania, po którym następuje n kroków uśredniania, możemy najpierw dokonać podwajania wierszy i kolumn, a następnie n -krotnie dokonać uśredniania wierszy i kolumn jednocześnie.

128

Mamy zatem płat B-sklejany

$$s(u, v) = \sum_i \sum_k \epsilon_{ik} N_i^n(u) N_k^n(v) = \sum_i \sum_k \mathbf{d}_{ik} M_i^n(u) M_k^n(v),$$

przy czym znamy punkty kontrolne ϵ_{ik} , reprezentujące płat w bazie tensorowej $\{N_i^n \otimes N_k^n; i, k \in \mathbb{Z}\}$, w której funkcje N_i^n są oparte na ciągu węzłów równoodległych. Chcemy znaleźć punkty kontrolne \mathbf{d}_{ij} reprezentujące ten sam płat w bazie określonej przez funkcje M_i^n oparte na dwukrotnie gęstszym ciągu węzłów równoodległych.

W kroku **podwajania** przyjmujemy

$$\mathbf{d}_{2i,2k}^{(0)} = \mathbf{d}_{2i+1,2k}^{(0)} = \mathbf{d}_{2i+1,2k+1}^{(0)} = \epsilon_{ik}.$$

W j -tym kroku **uśredniania**, dla $j = 1, \dots, n$, obliczamy

$$\mathbf{d}_{ik}^{(j)} = \frac{1}{4} (\mathbf{d}_{i-1,k-1}^{(j-1)} + \mathbf{d}_{i,k-1}^{(j-1)} + \mathbf{d}_{i-1,k}^{(j-1)} + \mathbf{d}_{i,k}^{(j-1)}).$$

Na końcu otrzymujemy $\mathbf{d}_{ik} = \mathbf{d}_{ik}^{(n)}$.

129

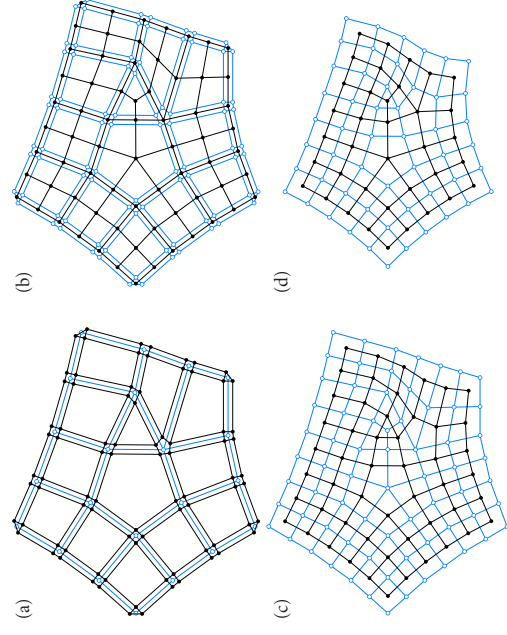
Ten algorytm jest łatwy do zaimplementowania, bo punkty kontrolne możemy trzymać w prostokątnej tablicy. W siatce możemy wyróżnić **wierzchołki** (to są punkty kontrolne), **krawędzie** (to są pary wierzchołków $(\mathbf{d}_{i-1,k}^{(j)}, \mathbf{d}_{i,k}^{(j)})$ oraz $(\mathbf{d}_{i,k-1}^{(j)}, \mathbf{d}_{i,k}^{(j)})$) i **ściany** (tj. czwórki punktów $(\mathbf{d}_{i-1,k-1}^{(j)}, \mathbf{d}_{i,k-1}^{(j)}, \mathbf{d}_{i-1,k}^{(j)}, \mathbf{d}_{i,k}^{(j)})$).

Podwajanie jest zastąpieniem każdej kolumny przez jej dwie kopie i każdego wiersza przez jego dwie kopie. Wtedy dla każdej krawędzi i dla każdego wierzchołka w siatce pojawia się nowa ściana czworokątna.

Uśrednianie jest obliczeniem wierzchołka w środku ciężkości każdej ściany.

Krawędzie otrzymanej w ten sposób siatki odpowiadają wspólnym krawędziom siatki danej, a ściany nowej siatki odpowiadają tym wierzchołkom siatki danej, z których wychodzą 4 krawędzie. Patrząc na siatkę jako na graf, widzimy, że uśrednianie jest konstrukcją fragmentu grafu dualnego.

130



132

Algorytm zagęszczania możemy zastosować do **siatek nierównolamanych**. Taka siatka ma **wierzchołki**, **krawędzie** i **ściany**, przy czym krawędź jest parą wierzchołków, a ściana jest zamkniętą lamana zbudowaną z krawędzi. Przy tym każda ściana ma różne wierzchołki, a każda krawędź należy do jednej albo dwóch różnych ścian. i dwie ściany mogą mieć co najwyżej jedną wspólną krawędź. Dodatkowo każdy wierzchołek wewnętrzny (taki, którego wszystkie krawędzie należą do dwóch ścian) ma co najmniej 3 wychodzące z niego krawędzie.

Taka siatka jest przybliżeniem pewnej powierzchni, która jest granicą ciągu siatek otrzymanych przez wielokrotne zagęszczanie.

131

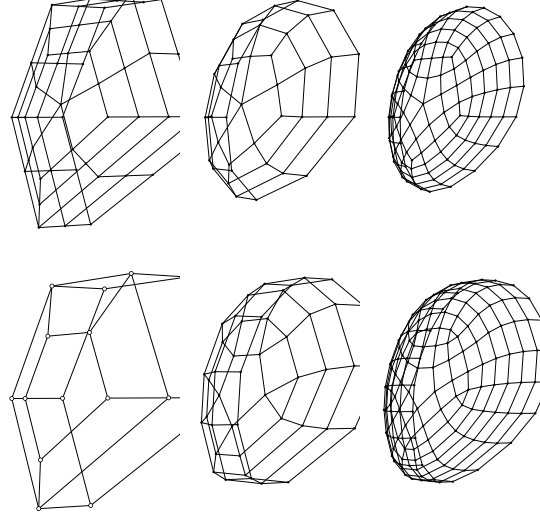
Powierzchnia graniczna składa się z przeliczalnie wielu płatów wielomianowych stopnia (n, n) , których parametryzacje są połączone z zachowaniem ciągłości pochodnych rzędu $(n - 1)$.

Elementy specjalne siatki to ściany nie-czworokątne i wierzchołki wewnętrzne, z których wychodzi inna niż 4 liczba krawędzi. Po podwajaniu wszystkie elementy specjalne są ścianami, uśrednianie przetwarza każdy wierzchołek wewnętrzny na ścianę o tylu samo krawędziach, a każdą ścianę, której wszystkie krawędzie są wewnętrzne, na wierzchołek z taką samą liczbą krawędzi.

W rezultacie liczba elementów specjalnych kolejnych siatek nie rośnie; jeśli n jest nieparzyste, to wszystkie elementy specjalne siatki są wierzchołkami, a jeśli parzyste, to ścianami.

Dwa najczęściej stosowane warianty tego zagęszczania są nazywane algorytmami Doo-Sabina (dla $n = 2$) i Catmulla-Clarka (dla $n = 3$).

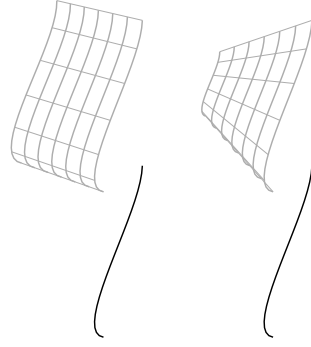
133



134

Modelowanie powierzchni i brył

Zakreślanie (*sweeping*) polega na przesuwaniu krzywej lub powierzchni (tzw. przekroju) wzdłuż odcinka (prowadnicy). Można przy tym dokonywać skalowania.



135

Możliwości uogólnienia zakreślenia są takie:

- Prowadnica może być krzywą.
- Każdemu punktowi prowadnicy może odpowiadać inne przekształcenie przekroju.
- Można dopuścić zmiany kształtu przekroju podczas „przesuwania go”.

136

Uogólnienie zakresłania, gdy przekrój jest krzywą, daje powierzchnię parametryczną określoną wzorem

$$s(u, v) = \mathbf{p}(u) + \mathbf{x}_1(u)x_q(v) + \mathbf{x}_2(u)y_q(v) + \mathbf{x}_3(u)z_q(v).$$

Tu jest 5 krzywych: **prownadnica** $\mathbf{p}(u)$, 3 **kierownice**, $\mathbf{x}_1(u)$, $\mathbf{x}_2(u)$, $\mathbf{x}_3(u)$ oraz **przekrój** $\mathbf{q}(v) = (x_q(v), y_q(v), z_q(v))$. Zauważmy, że możemy ten wzór przedstawić w postaci

$$\begin{bmatrix} s(u, v) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1(u) & \mathbf{x}_2(u) & \mathbf{x}_3(u) & \mathbf{p}(u) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{q}(v) \\ 1 \end{bmatrix},$$

a zatem dla każdego u odpowiednie punkty kierownicy i prownadnic okreśłają przekształcenie afiniczne przekroju.

Przypuścmy, że wszystkie krzywe są B-sklejane i prownadnica oraz kierownice są reprezentowane w tej samej bazie (tj. mają ten sam stopień n i te same węzły):

$$\mathbf{p}(u) = \sum_i \mathbf{p}_i N_i^n(u),$$

$$\mathbf{x}_k(u) = \sum_i \mathbf{x}_{ki} N_i^n(u), \quad k = 1, 2, 3,$$

$$\mathbf{q}(v) = \sum_j \mathbf{q}_j N_j^m(v).$$

Podstawiając te wyrażenia do wzoru opisującego powierzchnię zakreślaną, dostaniemy siatkę kontrolną powierzchni B-sklejanej $s(u, v)$:

$$s(u, v) = \sum_i \sum_j \mathbf{d}_{ij} N_i^n(u) N_j^m(v), \quad \begin{bmatrix} \mathbf{d}_{ij} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{1i} & \mathbf{x}_{2i} & \mathbf{x}_{3i} & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{q}_j \\ 1 \end{bmatrix}.$$

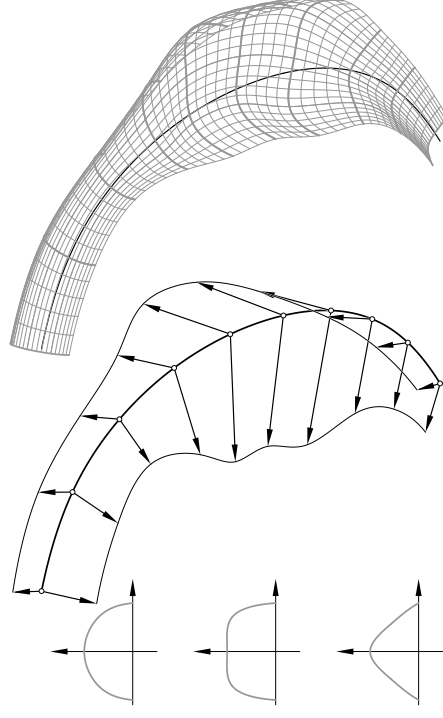
Mając procedurę wyświetlania powierzchni B-sklejanych, możemy utworzyć siatkę kontrolną powierzchni s i ją rysować, ale jeśli potrzebne są punkty tej powierzchni, to taniej jest obliczać odpowiednie punkty krzywych i wyznaczać punkty powierzchni na ich podstawie.

Największe uogólnienie zamiatania, gdy przekrój zmienia się podczas zamiatania, daje wzór

$$s(u, v) = \mathbf{p}(u) + \mathbf{x}_1(u)x_q(u, v) + \mathbf{x}_2(u)y_q(u, v) + \mathbf{x}_3(u)z_q(u, v).$$

Mamy tu jednoparametrową rodzinę krzywych parametrycznych

$\mathbf{q}(u, v) = (x_q(u, v), y_q(u, v), z_q(u, v))$, czyli formalnie płat powierzchni. Jeśli to jest płat B-sklejany, to znalezienie reprezentacji B-sklejanej płata s jest trudniejsze — wymaga mnożenia funkcji B-sklejanych tej samej zmiennej. Ale punkty tej powierzchni można wyznaczyć na podstawie punktów prownadnicy i kierownic oraz punktów $\mathbf{q}(u, v)$. Można też skonstruować rozwiązanie przybliżone, czyli powierzchnię rozpinaną.



Jeśli krzywe e i m są wymierne, reprezentowane przez krzywe jednorodnie

$$E(u) = \sum_i \begin{bmatrix} X_{ei} \\ Y_{ei} \\ W_{ei} \end{bmatrix} N_i^n(u), \quad M(v) = \sum_j \begin{bmatrix} X_{mj} \\ Y_{mj} \\ W_{mj} \end{bmatrix} N_j^m(v),$$

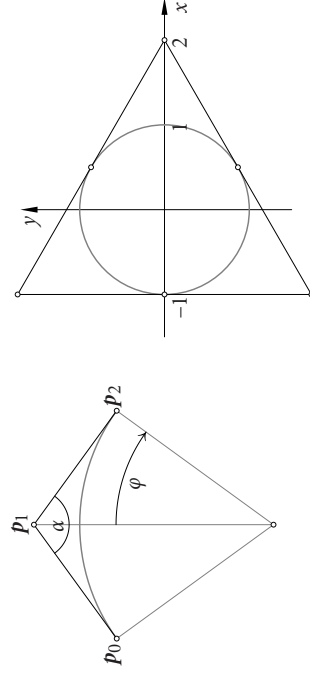
to punkty kontrolne płata jednorodnego reprezentującego iloczyn sferyczny można obliczyć ze wzoru

$$P_{ij} = \begin{bmatrix} X_{ei} X_{mj} \\ Y_{ei} X_{mj} \\ W_{ei} Y_{mj} \\ W_{ei} W_{mj} \end{bmatrix}.$$

To może się przydać, jeśli chcemy otrzymać powierzchnię obrotową, bo okrąg nie ma parametryzacji wielomianowej, ale ma parametryzację wymierną.

Łuk okręgu odpowiadający kątowi 2φ może być reprezentowany jako wymierna krzywa Béziera stopnia 2, której łamana kontrolna ma dwa odcinki o tej samej długości połączone pod kątem $\alpha = \pi - 2\varphi$. Środek okręgu jest punktem przecięcia prostych prostopadłych do tych odcinków, wystawionych w punktach końcowych łamanej, P_0 i P_2 . Wagi tych dwóch punktów powinny być równe 1 i wtedy waga punktu P_1 musi być równa $\cos \varphi$.

Cały okrąg możemy otrzymać jako połączenie trzech łuków o tej samej długości; dla każdego z tych łuków jest $\varphi = \frac{\pi}{3} = 60^\circ$ oraz $\cos \varphi = \frac{1}{2}$.



Reprezentacje scen trójwymiarowych

Obiekty trójwymiarowe dzielimy na

- obiekty z zamkniętą objętością (bryły),
- obiekty z otwartą objętością (krzywe, powierzchnie),
- obiekty objętościowe, częściowo przezroczyste (chmury, dym, płomień, futro).

Obiekty z zamkniętą objętością są obrazowane przez narysowanie powierzchni — brzęgu bryły. Obserwator widzi tylko jedną stronę tej powierzchni.

Obiekty dzielą się też na **prymitywy** i **obiekty złożone**; te pierwsze możemy opisać bezpośrednio, te opis tych drugich powstaje z przetworzonych prymitywów.

Reprezentacja sceny złożonej z wielu obiektów jest strukturą danych, która może służyć do wielu celów:

- rysowania sceny,
- obliczeń globalnego oświetlenia,
- wzmocnienia danych (np. generowania szczegółów),
- rozmieszczania obiektów w przestrzeni w określony sposób,
- symulacji ruchu,
- wykrywania kolizji,
- zmieniania stanu poszczególnych obiektów i ich konfiguracji w przestrzeni.

Jedna struktura danych może nie być odpowiednia do wszystkich tych celów i dlatego jest jeszcze jedno zadanie: wytworzenie ze struktury podstawowej struktur wyspecjalizowanych do poszczególnych celów.

149

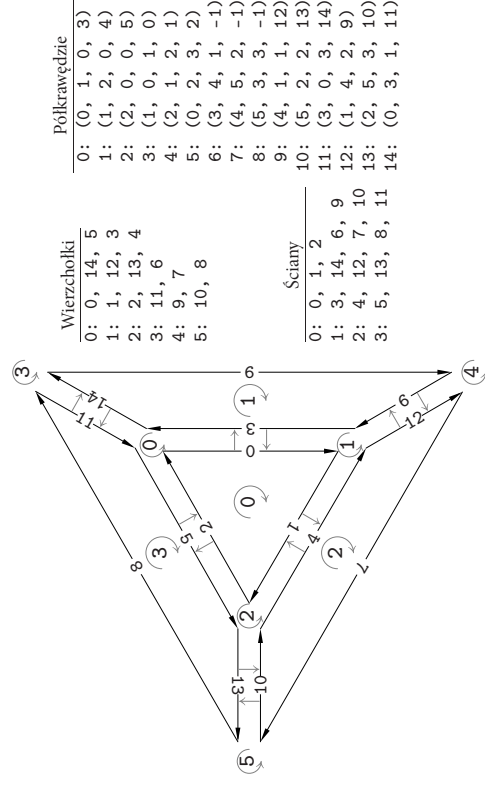
Bryła wielościenne jest obiektem z zamkniętą objętością, której brzeg składa się z płaskich wielokątów. Dowolny wielokąt może być podzielony na trójkąty, a zatem do narysowania bryły wystarczy lista trójkątów. Ale do innych celów może być potrzebna bardziej szczegółowa reprezentacja takiej bryły — w szczególności informacja o tym, które ściany sąsiadują ze sobą.

Reprezentacja brzegowa bryły składa się z tablic **wierzchołków**, **krawędzi** i **ścian**.

Wierzchołek ma określone położenie oraz inne atrybuty, np. wektor normalny, kolor; współrzędne tekstury. Może mieć też listę (identyfikatorów) wychodzących z niego krawędzi.

Krawędź ma identyfikatory wierzchołków oraz ścian, które rozgranicza. Wygodnie jest reprezentować krawędź za pomocą pary **półkrawędzi** — każda półkrawędź jest zorientowana, tj. ma wyróżniony początek i koniec i należy tylko do jednej ściany. W każdej półkrawędzi jest przechowywany identyfikator jej „drugiej połowy”, zorientowanej przeciwnie.

150



152

Zorientowanie półkrawędzi jest zaletą, ponieważ łatwiej jest wyszukiwać odpowiednie informacje. Na przykład łatwo jest obejść ścianę po jej półkrawędziach, zgodnie z ich orientacją.

Ściana może zawierać listę (identyfikatorów) wierzchołków lub krawędzi (albo półkrawędzi), może mieć też atrybuty dodatkowe, np. wektor normalny.

Można dopuścić ściany będące dowolnymi wielokątami (także niejednostajnymi), co bardzo komplikuje reprezentację. Ograniczenie do wielokątów jednostajnych (tj. bez otworów) umożliwia proste przechowywanie identyfikatorów krawędzi (lub półkrawędzi) w pojedynczej tablicy.

151

Konstrukcyjna geometria brył

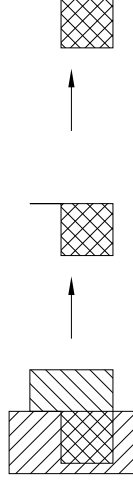
Prostopadłościan, graniastosłup lub ostrosłup, a także kula, walec lub stożek mogą być prymitywami, ponieważ ich opis jest dość prosty. Obiekty bardziej skomplikowane mogą mieć opisy zbyt skomplikowane, aby je wykonywać „ręcznie”, w związku z czym do wytworzenia tych opisów zatrudnia się komputery: na prymitywach i obiektach uzyskanych z nich wcześniej wykonuje się operacje mnogościowe, których wynikiem mogą być obiekty całkiem skomplikowane.

Operacje mnogościowe to dopełnienie, suma, różnica, przecięcie i różnica symetryczna; umiając znaleźć dopełnienie i sumę albo przecięcie, możemy konstruować wyniki pozostałych działań.

Konstrukcyjna geometria brył (*constructive solid geometry, CSG*) jest uzupełnieniem operacji mnogościowych na figurach geometrycznych o tzw. **regularyzację**. Matematycznie jest to domknięcie wnętrza figury. Powstają w ten sposób **bryły regularne**; w otoczeniu każdego punktu na brzegu takiej bryły są punkty położone wewnątrz niej i punkty do niej nienależące.

153

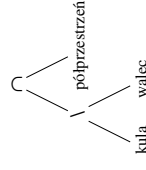
Regularyzacja przecięcia figur płaskich:



Regularyzacja brył powoduje odrzucenie wszystkich izolowanych punktów, krzywych i powierzchni nieprzylegających do wnętrza bryły.

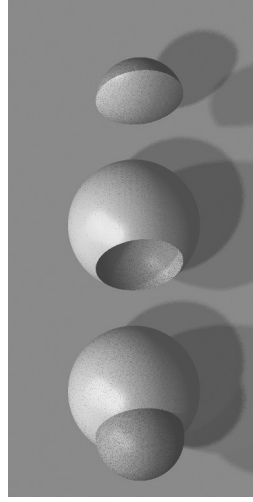
154

Bardziej skomplikowane bryły opisuje się za pomocą wyrażeń mnogościowych z wieloma operacjami, przy czym można je przedstawiać w postaci drzewa binarnego — drzewo takie może być zaimplementowane jako struktura danych reprezentująca wyrażenie.



155

Elementarne operacje CSG — sumę, różnicę i przecięcie pokazuje rysunek:



156

Reprezentacja brzegowa wielościanu za pomocą siatki zbudowanej z wierzchołków, półkrawędzi i ścian umożliwia łatwe znalezienie **dopełnienia**. Ponieważ brzeg jest zorientowany (bo półkrawędzie są zorientowane), wektory normalne ścian wypukłych obliczone jako iloczyn wektorowe półkrawędzi są zorientowane na zewnątrz (albo do wewnątrz) bryły. Wystarczy zatem odwrócić orientację wszystkich krawędzi.

Regularyzacja w tym przypadku nie wymaga wykonywania żadnych obliczeń.

157

Aby znaleźć **przecięcie** dwóch brył, trzeba wykonać algorytm będący rozszerzeniem na trzy wymiary algorytmu Weilera–Athertona.

1. Dla każdej pary ścian, z których jedna jest częścią pierwszej, a druga drugiej bryły, trzeba znaleźć przecięcie tych ścian. Może ono być zbiorem pustym, punktem, odcinkiem, wielokątem lub sumą punktów, odcinków i wielokątów. Jeśli przecięcie zawiera wielokąt, to znajdujemy odcinki — części wspólne krawędzi jednej ściany z drugą.
2. Odcinkami znanymi w pierwszym kroku dzielimy ściany bryły na spójne fragmenty. Wnętrze każdego fragmentu leży w całości wewnątrz, na brzegu lub na zewnątrz drugiej bryły. Określamy **graf sąsiedztwa ścian** — jego wierzchołkami są fragmenty ścian, a krawędziami krawędzie ścian lub odcinki znalezione w pierwszym kroku.

158

Reprezentacja brzegowa wielościanu za pomocą siatki zbudowanej z wierzchołków, półkrawędzi i ścian umożliwia łatwe znalezienie **dopełnienia**. Ponieważ brzeg jest zorientowany (bo półkrawędzie są zorientowane), wektory normalne ścian wypukłych obliczone jako iloczyn wektorowe półkrawędzi są zorientowane na zewnątrz (albo do wewnątrz) bryły. Wystarczy zatem odwrócić orientację wszystkich krawędzi.

Regularyzacja w tym przypadku nie wymaga wykonywania żadnych obliczeń.

157

3. Regularyzacja polega na odrzuceniu wierzchołków grafu reprezentujących fragmenty ścian nieprzylegające do wnętrza przecięcia brył.

4. Wybieramy nieodwiedzony wierzchołek grafu (fragment ściany) i sprawdzamy, czy on należy do brzegu przecięcia.

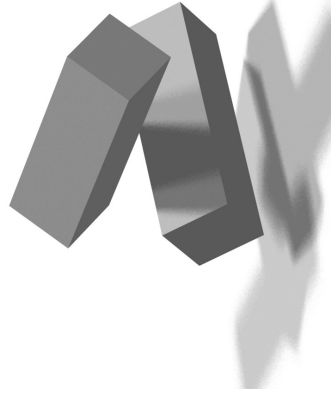
5. Jeśli tak, to metodą DFS lub BFS przeszukujemy graf sąsiedztwa ścian, wprowadzając wierzchołki należące do brzegu przecięcia.

Kroki 3 i 4 powtarzamy, do chwili, gdy wszystkie wierzchołki zostaną odwiedzone.

6. Może być jeszcze potrzebny postprocesing, np. podział ścian na fragmenty jednocspójne.

159

Najwięcej kłopotów sprawia uodpornienie implementacji algorytmu na przypadki szczególne, np. takie jak na rysunku.



160

Drzewa i grafy scen

Reprezentacja sceny powinna uwzględniać **hierarchię obiektów**. Do wykonania obrazów w zasadzie wystarczy liniowa lista obiektów do wyświetlenia, ale w aplikacjach graficznych są też inne potrzeby.

Po pierwsze obiekty bywają zbudowane z pewnych części i same składają się w większe zespoły, którymi można manipulować jako całościami. Naturalną reprezentacją hierarchii jest drzewo, którego korzeń reprezentuje całą scenę, a liście — poszczególne obiekty.

161

Wierzchołki drzewa mogą mieć następujące atrybuty:

Przekształcenie geometryczne opisujące przejście między układami współrzędnych związanymi z danym wierzchołkiem i wierzchołkiem drzewa wyżej. Przekształcenie związane z korzeniem może reprezentować przejście do układu współrzędnych świata.

Można łączyć przekształcenia geometryczne z krawędziami drzewa, nie tylko z wierzchołkami. To ułatwia realizację **łańcuchów kinematycznych**, w których te przekształcenia są animowane.

Bryła otaczająca, na przykład kula lub prostopadłościenna kostka. Kostkę łatwo jest znaleźć dla brył wielościennych lub dla płatów B-sklejanych (z własności otoczki wypukłej).

Bryła otaczająca przydaje się podczas rysowania — jeśli jest w całości niewidoczna, to niewidoczne są też obiekty w niej zawarte i nie trzeba ich rysować. Można też oszczędzać czas, nie wykrywając kolizji między obiektami zawartymi w rozłącznych bryłach otaczających.

162

Uproszczona reprezentacja obiektu, którą można rysować, gdy obraz obiektu jest mały — wyboru można dokonać po zbadaniu wielkości obrazu bryły otaczającej.

Uproszczenia mogą być obecne na wielu poziomach drzewa, na przykład na najwyższym poziomie model samochodu może być poteksturowanym prostopadłościaniem, na poziomie średnim koło samochodu może być walcem, a najdokładniejsza reprezentacja koła, do rysowania z bardzo bliska, będzie miała wszystkie szprychy, wentyle i bieżnik na oponach.

Działanie CSG. Wierzchołek drzewa może przechowywać identyfikator operacji CSG, którą trzeba wykonać na obiektach reprezentowanych przez korzenie poddrzew. To jest dopuszczalne, jeśli algorytm wizualizacji dopuszcza taką reprezentację obiektów, najzwyklej może to być algorytm śledzenia promieni. Jeśli używamy algorytmu z buforem głębokości (np. w OpenGL-u), to reprezentację brył CSG w postaci drzewa raczej rozdzielimy z drzewem hierarchii sceny, co za dużo, to niedługo.

163

Pewne obiekty występują w scenach „w wielu egzemplarzach”. Samochód ma zazwyczaj cztery jednakowe koła. W scenie może być kilka samochodów tej samej marki itd.

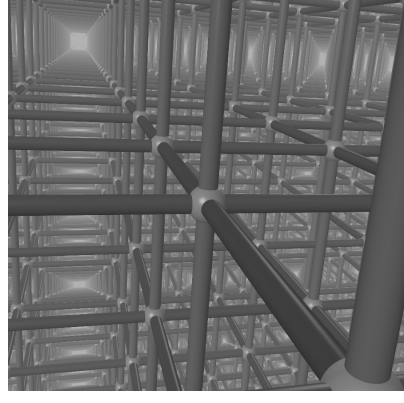
Bezcyklowy graf skierowany (*directed acyclic graph*, DAG) może być uogólnieniem drzewa hierarchii sceny. Jego zaletą jest zmniejszenie pamięci zajmowanej przez opis sceny i możliwość powielania klocek, z których scena jest zbudowana, bez potrzeby opisywania każdego klocka osobno.

Graf skierowany sceny ma korzeń, od którego podczas rysowania sceny zaczyna się przeszukiwanie metodą DFS, przy czym należy przejść *wszystkie* ścieżki w grafie od korzenia do liści (lub do wierzchołków z wybraną reprezentacją uproszczoną obiektów). Macierze przekształceń geometrycznych wzdłuż ścieżek wyznaczamy, otrzymując dla każdego liścia odpowiednią macierz przekształcenia modelu.

164

Opis sceny za pomocą DAG można sparametryzować. Parametr może być kolorem lakieru samochodu — wtedy każdy egzemplarz samochodu może być w innym kolorze. Można też wprowadzić parametry artykulacji łańcucha kinematycznego. Poszczególne samochody będą wtedy mogły mieć inaczej skrecone koła lub pootwierane drzwi.

165



Obrazek przedstawia scenę zbudowaną z 2^{16} kul i $3 \cdot 2^{11}$ walców. DAG reprezentujący tę scenę miał 93 wierzchołki, przy czym wszystkie wierzchołki oprócz korzenia i liści miały wskaźniki do dwóch wierzchołków na następnym poziomie.

166

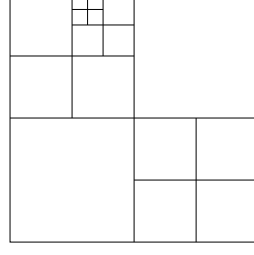
Drzewa binarne, czwórkowe i ósemkowe

Złożoność wielu zadań geometrii obliczeniowej zależy od rozmieszczenia przetwarzaných obiektów w przestrzeni — różne algorytmy rozwiązujące takie zadania są tym skuteczniejsze, im lepiej „rozdzielają” obiekty, aby w każdym momencie obliczeń móc przetwarzać tylko niewiele z nich naraz.

Jedną z możliwości to korzystanie z brył otaczających w drzewie hierarchii sceny. Inną możliwość to wprowadzenie takiej hierarchii przez taki podział przestrzeni na komórki, aby do każdej komórki trafiła tylko część obiektów.

167

Drzewo czwórkowe (*quadtree*) jest często używane w zadaniach płaskich. Jeśli przetwarzane obiekty znajdują się w obszarze ograniczonym, to na tym obszarze opisuje się prostokąt, reprezentowany przez korzeń drzewa. Następnie dokonuje się rekurencyjnego podziału prostokąta na cztery przystające części (prostokątne boksy) — adaptacyjnie, tj. podejmując decyzję o podziale na podstawie zbioru obiektów przecinających się z prostokątem.



168

Zastosowania:

- Przeszukiwanie obszarów (np. w systemach informacji przestrzennej).
- Konstrukcyjna geometria figur płaskich.
- Algorytmy widoczności.
- Wykrywanie kolizji obiektów na płaszczyźnie.

169

Trójwymiarowym odpowiednikiem drzew czwórkowych są **drzewa ósemkowe** (*octrees*), w których rekurencyjnemu podziałowi podlega prostopadłościenna kostka otaczająca obszar z przetwarzanymi obiektami. Każdy wierzchołek takiego drzewa, który nie jest liściem, ma 8 wskaźników do poddrzew reprezentujących prostopadłościenne boksy. Taka struktura danych jest często używana w algorytmie śledzenia promieni, do ograniczania złożoności obliczeniowej algorytmu wyznaczania przecięć promieni z obiektami.

170

Typowe podzadania rozwiązywane przy użyciu drzew czwórkowych lub ósemkowych to:

- Przeszukanie całego drzewa i przetworzenie „każdy z każdym” obiektów występujących w listach obiektów przecinających się z boksami.
- Wyszukanie wierzchołków drzewa reprezentujących boksy zawierające podany punkt.
- Znalezienie boksów przecinających się z pewną figurą (w szczególności półprostą — w śledzeniu promieni).

To wymaga znajdowania wierzchołków reprezentujących boksy sąsiadujące — przez krawędź lub przez ścianę.

171

Własnością drzew czwórkowych lub ósemkowych jest to, że sąsiadujące obszary reprezentowane przez liście drzewa albo mają wspólną krawędź (ścianę), albo krawędź (ściana) jednego z tych obszarów jest częścią krawędzi (ściany) drugiego obszaru. To umożliwia wiele różnych strategii przeszukiwania drzewa:

- Wierzchołki zawierające podany punkt można wyszukiwać zawsze od korzenia.
- Jeśli trzeba znaleźć obszar sąsiedni, to od bieżącego wierzchołka można iść w górę, do znalezienia najmniejszego na tej drodze boksu zawierającego punkt „tuż za” wspólną ścianą, a potem w dół.
- Można znaleźć najmniejszy boks zawierający dany punkt, za pomocą kodu Mortona, a następnie iść tylko w górę (tj. w kierunku korzenia).

Trzeba pamiętać, że żadna z tych strategii nie jest istotnie lepsza od pozostałych; wiele zależy od budowy drzewa, zdefiniowanej przez rozkład obiektów w przestrzeni.

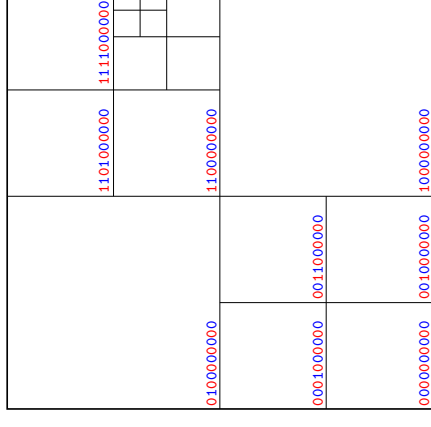
172

Kod Mortona w kostce d -wymiarowej jest zdefiniowany następująco: dla punktu p , którego współrzędne mają wartości z przedziału $[0, 1)$, d najbardziej znaczących bitów to są najbardziej znaczące bity (części ułamkowych) kolejnych współrzędnych przedstawionych w układzie dwójkowym, kolejne d najbardziej znaczących bitów to są bity tych współrzędnych na następnej pozycji itd.

Długość kodu trzeba ograniczyć — do iloczynu wysokości drzewa i wymiaru kostki. Liczby całkowite 32-bitowe umożliwiają określenie kodów Mortona dla drzew ośmiokowych o wysokości 10. W każdym wierzchołku drzewa możemy zapamiętać kod Mortona odpowiadający narożnikowi boksu najbliższemu punktu $(0, 0, 0)$ i utworzyć tablicę wskaźników do liści drzewa, uporządkowaną w kolejności rosnących kodów. Mając dany punkt, utworzymy jego kod Mortona i zastosujemy wyszukiwanie binarne w tej tablicy.

173

Przykład: drzewo czwórkowe ma wysokość 5, kody są 10-bitowe.



174

Większą elastyczność sposobu dzielenia kostki dają k - d drzewa, które są drzewami binarnymi; na każdym poziomie boks jest dzielony na dwa prostopadłościennne boksy, przy czym nie muszą one mieć jednakowych wymiarów. Na każdym poziomie drzewa jest ustalony kierunek podziału boksu, np. płaszczyzną $z = \text{const}$, $x = \text{const}$ albo $y = \text{const}$, w wierzchołku drzewa zapamiętuje się const wybrane tak, aby jak najlepiej dostosować się do rozmieszczenia obiektów w boksie; jeśli obiekty są punktami, to granica podziału może dzielić ich zbiór na połowy.

Wadą k - d drzew jest to, że wspólne brzegi bokśw nie pasują do siebie tak dokładnie, jak brzegi bokśw w drzewach czwórkowych i ośmiokowych, co ogranicza wybór algorytmów przechodzenia do sąsiednich bokśw. Również kody Mortona nie mają tu zastosowania.

175

Najbardziej ogólny sposób dzielenia przestrzeni płaszczyznami na komórki umożliwiają **drzewa binarne podziału przestrzeni** (*binary space partition trees*, *BSP-trees*), których również można użyć do podziału przestrzeni o dowolnym wymiarze. Korzeń drzewa reprezentuje całą przestrzeń i jest w nim przechowywana reprezentacja (współczynniki równania) hiperpłaszczyzny podziału. Reprezentacja ta zawiera wektor normalny hiperpłaszczyzny, który ma określony zwrot. Jedno z poddrzew reprezentuje dalszy podział przestrzeni po stronie „in”, a drugie po (wskazywanej przez wektor normalny) stronie „out” hiperpłaszczyzny dzielącej.

176

Mając ściany, tzn. odcinki na płaszczynie lub płaskie wielokąty w przestrzeni trójwymiarowej, na ogół wybiera się hiperpłaszczyzny dzielące, w których leżą te figury, po ustaleniu ich kolejności. Pierwsza ściana określa zatem podział przestrzeni na dwie półprzestrzenie. Każda następna ściana albo znajduje się po jednej lub drugiej stronie hiperpłaszczyzny, albo ją przecina. Takie ściany trzeba podzielić na dwie części, do czego przydają się algorytmy obcinania przedstawione na wykładach wcześniej.

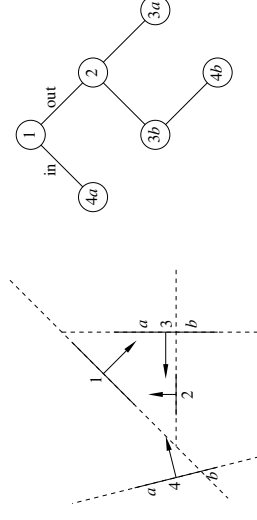
Ściana lub jej obcięta część służy do rozbudowania drzewa BSP tylko po jednej stronie — a zatem trafia do wierzchołka drzewa, który reprezentuje dalszy podział komórki przestrzeni na dwie części.

177

- Koszt budowy drzewa jest nie mniejszy niż $O(n \log n)$ operacji (jeśli drzewo jest idealnie zrównoważone i żadnej ściany nie trzeba dzielić) i nie większy niż $O(n^3)$ operacji (jeśli każda ściana przecina się z hiperpłaszczyznami wszystkich innych ścian) — niezależnie od kolejności otrzymamy $\frac{1}{2}(n^2 + n)$ fragmentów ścian.
- Jeśli ściany są ścianami wielościanu wypukłego, to dla każdej ściany wszystkie pozostałe ściany są po jej jednej stronie („in”) i wysokość drzewa BSP jest równa liczbie ścian. Z tego powodu można dołożyć pewną liczbę dodatkowych płaszczyn podziału, aby zbudować drzewo bardziej zrównoważone (i niższe).
- Dobierając kolejność ścian, najlepiej jest w każdym kroku wybrać ścianę, która nie podzieli pozostałych ścian i która najrówniej podzieli zbiór ścian na podzbiory znajdujące się po obu stronach.

179

Przykład płaski:



Ściana 3 trafia w całości do komórki 1in, ale przecina prostą dzielącą tę komórkę (prostą, na której leży ściana 2). Dlatego została podzielona na części 3a i 3b. Podobny los spotkał ścianę 4.

178

Drzewo BSP może być użyte do następującego algorytmu widoczności:

Metodą DFS przeszukujemy drzewo, rysując napotkane w wierzchołkach fragmenty ścian w kolejności infiksowej; przetwarzając wierzchołek drzewa, procedura rekurencyjna najpierw rysuje ściany znajdujące się po przeciwnej stronie płaszczyny dzielącej, potem ścianę dzielącą, a na końcu ściany po tej samej stronie co obserwator.

Zapewnia to poprawny efekt na końcowym obrazie: ściana dzieląca może (nie musi) zasłonić ściany po przeciwnej stronie lub ich fragmenty, a sama może być zasłonięta tylko przez ściany położone po tej samej stronie co obserwator. Piksele zamalowane kolorem wcześniej rysowanych ścian mogą zmienić kolor. Jest to tzw. **algorytm malarza**, który zapewnia właściwy kolor każdego piksela na końcowym obrazie.

180

Algorytmy widoczności

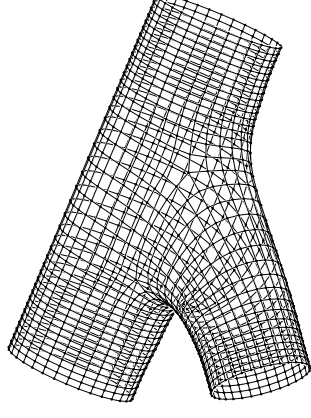
Znanych jest wiele algorytmów widoczności i nawet jeśli jeden z nich ma obecnie największe znaczenie, pozostałe algorytmy warto znać, bo mają one pewne możliwości, które dosyć trudno jest odtworzyć z tym jednym algorytmem.

Algorytmy widoczności dzielą się na

- **algorytmy przestrzeni danych** — wyznacza się w nich widoczne części obiektów do narysowania, można je potem zrzutować i narysować,
- **algorytmy przestrzeni obrazu** — wyznacza się w nich obraz, czyli kolory poszczególnych pikseli.

181

Dalej mamy podział na **algorytmy linii zasłoniętych** i **algorytmy powierzchni zasłoniętych**. W pierwszym przypadku zadanie polega na narysowaniu widocznych części krawędzi nieprzezroczystych wielokątów i innych odcinków rozmieszczonych w przestrzeni na tych wielokątach lub między nimi. Może być też tak, że scena składa się z samych odcinków.



182

Ważną cechą każdego algorytmu jest dopuszczalna klasa danych.

- Tylko powierzchnie płaskie (np. wielokąty) lub także powierzchnie zakrzywione.
- Zbiory powierzchni, które mogą się przecinać, lub mieć co najwyżej wspólne brzegi.
- Obiekty z otwartą objętością (powierzchnie, których obie strony mogą być widoczne) lub tylko brzegi brył.
- Bryły o jawnie wyznaczonej reprezentacji brzegu lub drzewa CSG.
- Tylko powierzchnie lub także obiekty objętościowe (chmury itp.).

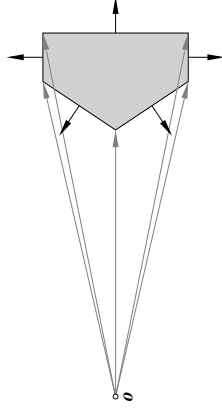
183

Im bardziej ogólne dane dopuszcza algorytm, tym bardziej jest skomplikowany i tym bardziej może być zawodny. Ponadto tym trudniejsza jest jego implementacja, w szczególności równoległa. Dlatego algorytmy implementowane w sprzęcie stosują metodę „brutalnej siły”, ograniczając klasę dopuszczalnych danych (do punktów, odcinków i trójkątów) i wymuszając przybliżanie np. powierzchni zakrzywionych dużymi ilościami małych trójkątów, nawet jeśli teoretycznie algorytm z buforem głębokości (i inne) może działać dla powierzchni zakrzywionych.

184

Algorytmy przestrzeni danych

Pewne algorytmy przestrzeni danych dają tylko rozwiązanie częściowe problemu widoczności, eliminując tylko niektóre niewidoczne obiekty w scenie. Takim algorytmem jest odrzucanie ścian obiektu z zamkniętą objętością „odwróconych tyłem” do obserwatora.



185

Przed rysowaniem obiektu trzeba wykonać instrukcje

```
glEnable ( GL_CULL_FACE );  
glCullFace ( GL_BACK );  
glFrontFace ( GL_CCW );
```

Pierwsza z nich włącza odrzucanie ścian, druga nakazuje odrzucanie ścian odwróconych tyłem, a trzecia określa, że odwrócone przodem są te trójkąty, które na obrazie mają wierzchołki uporządkowane w kierunku przeciwnym do ruchu wskazówek zegara.

Przed rysowaniem kolejnego obiektu, który nie jest bryłą, odrzucanie ścian trzeba wyłączyć, instrukcją

```
glDisable ( GL_CULL_FACE );
```

187

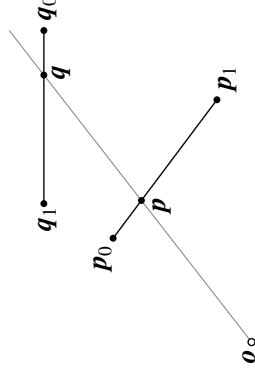
Jeśli scena składa się tylko z jednej wielościennej bryły wypukłej, to wyświetlenie tylko ścian „odwróconych przodem” daje pełne rozwiązanie zadania, ale to rzadko wystarczy w praktyce. Ale odrzucenie ścian odwróconych tyłem zmniejsza średnio o połowę liczbę ścian, których widoczność trzeba dalej rozstrzygnąć innym sposobem, więc to się oplaca.

W aplikacji OpenGL-a trzeba zadbać o to, aby wszystkie trójkąty w brzegu bryły miały zgodną orientację — jeśli trójkąt ma wierzchołki p_i , p_j i p_k (podane w takiej kolejności, dotyczy to pierwszego trójkąta w każdej taśmie lub wachlarzu), to wektor normalny $n = (p_j - p_i) \wedge (p_k - p_i)$ musi być zorientowany (dla ustalenia uwagi) na zewnątrz bryły.

Dalej zakładamy, że obserwator znajduje się na zewnątrz bryły i iloczyn macierzy VM (opisujący przejście od układu modelu do układu obserwatora) ma dodatni wyznacznik.

186

Ważnym krokiem algorytmów przestrzeni danych — linii i powierzchni zasłoniętych — jest rozstrzygnięcie widoczności między dwoma odcinkami w przestrzeni.



188

Poszukiwane punkty p i q , które leżą na prostej przechodzącej przez położenie obserwatora o , można wyrazić tak:

$$p = p_0 + s(p_1 - p_0),$$

$$q = q_0 + t(q_1 - q_0).$$

Punkty o , p i q leżą na jednej prostej, jeśli istnieje liczba u , taka że $p - o = u(q - o)$. Stąd wynika układ równań

$$p_0 + s(p_1 - p_0) - o = u(q_0 + t(q_1 - q_0) - o).$$

Po uporządkowaniu dostajemy układ równań liniowych 3×3 :

$$[p_1 - p_0, o - q_0, q_0 - q_1]x = o - p_1$$

z niewiadomym wektorem $x = (s, u, t)$.

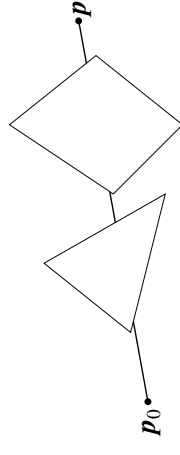
189

Interpretujemy wyniki:

- Rząd 0 macierzy układu nie jest możliwy, jeśli odcinki mają niezerową długość.
- Rząd 1 oznacza, że wszystkie kolumny macierzy mają ten sam kierunek. Jeśli układ jest niesprzeczny, to jeden odcinek może zasłaniać drugi, ale obrazem obu odcinków jest punkt. Jeśli odrzuciliśmy ściany, w których płaszczyźnie znajduje się obserwator (ściany widoczne jako odcinki), to taka sytuacja nie wystąpi.
- Jeśli macierz układu ma rząd 2 i układ jest niesprzeczny, to część jednego odcinka może zasłaniać część drugiego. Odcinki mogą się też przecinać. Aby to zbadać, trzeba znaleźć rozwiązanie dla $s = 0$, $s = 1$, $t = 0$ i $t = 1$ i zbadać wartości pozostałych niewiadomych.
- Jeśli macierz jest nieosobliwa, to zasłanianie ma miejsce, gdy $s, t \in [0, 1]$ oraz $u > 0$. Dokładniej, jeśli $u \in (0, 1)$, to punkt p zasłania punkt q , a jeśli $u > 1$, to punkt p jest zasłaniany przez q .

190

Każda kolejna ściana, jeśli to nie jest ściana, na której leży odcinek, może zasłonić jego część lub całość. Wypukłość ścian bardzo upraszcza obliczenia. Lista widocznych fragmentów może się wydłużać lub skrócić, może się też okazać, że cały odcinek jest niewidoczny.



Złożoność obliczeniową (rzędu liczby odcinków w scenie do kwadratu) można ograniczyć, jeśli dokona się rzutowania sceny na płaszczyznę i użyje technik geometrii obliczeniowej (zamiastania) w celu wyeliminowania par (odcinek,ściana) poddawanych obliczeniom.

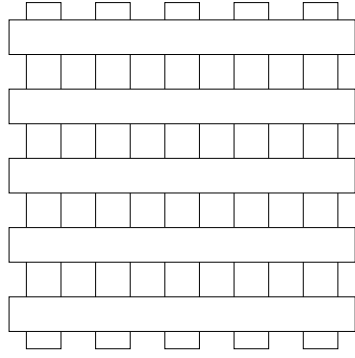
192

Algorytm Ricciego (1980 r.) jest algorytmem linii zasłoniętej dla sceny złożonej z wypukłych wielokątów, które mogą mieć co najwyżej wspólne krawędzie. Dane składają się ze zbioru ścian i zbioru krawędzi oraz innych odcinków, położonych na ścianach lub „luzem” w przestrzeni (wtedy nie mogą przecinać ścian), oraz położenia obserwatora.

Wynikiem algorytmu jest zbiór widocznych fragmentów tych odcinków. Algorytm używa strategii „każdy z każdym”, tj. sprawdza widoczność każdego odcinka z wszystkimi ścianami. Dla każdego odcinka tworzy się listę widocznych fragmentów (początkowo jednoelementową, element reprezentuje cały odcinek).

191

Kwadratowy koszt może jednak być optymalny z uwagi na złożoność opisu wyniku — w scenie zawierającej n odcinków może być $\Theta(n^2)$ widocznych fragmentów odcinków.



193

Algorytm Weilera–Athertona jest algorytmem powierzchni zasłoniętej w scenie złożonej z płaskich wielokątnych ścian. Użyta w tym algorytmie strategia „każdy z każdym” polega na przetwarzaniu wszystkich par ścian. Jedna ściana (położona bliżej) jest rzutowana na płaszczyznę drugiej ściany, po czym wyznaczana jest różnica wielokątów — jest to niezasłonięta część drugiej ściany. Używany jest do tego algorytm Weilera–Athertona obcinania, opisany w jednym z wcześniejszych wykładów.

Jeśli widoczność jest rozstrzygana z punktu położenia źródła światła, to algorytm wyznacza oświetlone fragmenty ścian, co można wykorzystać do otrzymania obrazu sceny z cieniami.

Rozszerzeniem tego pomysłu jest algorytm śledzenia wiązek (*beam tracing*) Paula Heckberta i Pata Hanrahana (1984 r.), w którym można było otrzymać obraz sceny, w której pewne wielokąty są lustrami. Obecnie ten efekt otrzymuje się, np. w OpenGL-u, przy użyciu tekstur, ale używane w tej technice konstrukcje geometryczne są takie same.

194

Algorytm Appela ma w założeniu mniejszą złożoność niż algorytm Ricciego dla scen o tej samej postaci. Widoczność odcinka zmienia się w punktach zasłanianych przez **krawędzie sylwetkowe**, których liczba zazwyczaj jest znacznie mniejsza niż liczba wszystkich odcinków w scenie. Krawędź sylwetkowa należy tylko do jednej ściany lub rozgranicza ściany, z których jedna jest odwrócona przodem do obserwatora, a druga nie jest.

Definiujemy **stopień zasłonięcia punktu** w przestrzeni — jest to liczba (odwróconych przodem) ścian zasłaniających dany punkt przed obserwatorem.

Wyznaczamy dla każdego odcinka punkty zasłaniane przez krawędzie sylwetkowe i dzielimy odcinki tymi punktami na fragmenty. Dla każdego takiego punktu znajdujemy informację o tym, jak stopień zasłonięcia zmienia się podczas przejścia przez ten punkt.

195

Odcinki w scenie tworzą pewien graf, którego są krawędziami. Punkty podziału wprowadzają nowe wierzchołki dzielące krawędzie, dzięki czemu każda krawędź ma stały stopień zasłonięcia. Należy teraz przeszukać graf i wyprowadzić wszystkie krawędzie, których stopień zasłonięcia jest równy 0. Mając spójną składową takiego grafu krawędziowego, wystarczy znaleźć stopień zasłonięcia dowolnej krawędzi, a następnie przeszukać tę składową, dodając przyrosty stopnia zasłonięcia w poszczególnych wierzchołkach.

Wadą algorytmu Appela jest to, że ewentualne błędy propagują się. Jeśli stopień zasłonięcia dowolnej krawędzi został błędnie wyznaczony, to wszystkie krawędzie tej składowej mogą być błędnie wyprowadzone lub nie wyprowadzone, a to wymaga dużej ilości pracy przy implementacji, aby algorytm działał poprawnie dla scen, w których wiele krawędzi sylwetkowych spotyka się w jednym wierzchołku i takie wierzchołki mogą być współliniowe z położeniem obserwatora.

196

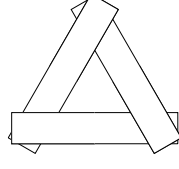
Algorytmy przestrzeni obrazu

Wynikiem działania tych algorytmów jest obraz rastrowy; jeśli położenie obserwatora nie zmienia się, ale zmienimy rozdzielczość obrazu lub kierunek patrzenia, to całe obliczenie trzeba powtórzyć.

Algorytm malarza polega na posortowaniu ścian (lub innych obiektów) w kolejności od najdalej od obserwatora do położonej najbliżej. W tej kolejności ściany należy zrasteryzować i wyświetlić. W ten sposób każdy piksel albo pozostanie w kolorze tła, albo jego końcowy kolor będzie kolorem punktu na ścianie zasłaniającej wszystkie ściany, których rzut zawiera ten piksel.

197

Kłopot polega na tym, że punkty ścian zazwyczaj mają głębokości wypełniające pewne przedziały. Relacja między ścianami o rozłącznych przedziałach podczas sortowania jest oczywista, ale jeśli przedziały przecinają się, to bez dodatkowych testów nie da się stwierdzić, która ściana zasłania drugą. Zresztą, ściany mogą się przecinać, a poza tym może nastąpić zakleszczenie, takie jak na rysunku.



Metodą sortowania, połączoną z dzieleniem ścian przecinających się lub powodujących zakleszczenie jest budowanie drzewa BSP, są też inne algorytmy dające równoważne skutki.

198

Algorytmy przestrzeni obrazu

Wynikiem działania tych algorytmów jest obraz rastrowy; jeśli położenie obserwatora nie zmienia się, ale zmienimy rozdzielczość obrazu lub kierunek patrzenia, to całe obliczenie trzeba powtórzyć.

Algorytm malarza polega na posortowaniu ścian (lub innych obiektów) w kolejności od najdalej od obserwatora do położonej najbliżej. W tej kolejności ściany należy zrasteryzować i wyświetlić. W ten sposób każdy piksel albo pozostanie w kolorze tła, albo jego końcowy kolor będzie kolorem punktu na ścianie zasłaniającej wszystkie ściany, których rzut zawiera ten piksel.

197

Algorytm z buforem głębokości jest tym najważniejszym w praktyce algorytmem. Jego zaletami są prostota i niezawodność oraz łatwość implementowania w sprzęcie. Przypomnijmy: dla każdego piksela mamy dodatkową zmienną liczbową, przechowywaną w tablicy zwanej buforem głębokości (*depth buffer* albo *z-buffer*).

Oczekiwany zakres głębokości obiektów w scenie jest odwzorowany na zakres wartości zmiennych w tym buforze. Przed przystąpieniem do rysowania obiektów wszystkim tym zmiennym jest przypisywana wartość odpowiadająca maksymalnej głębokości. Dla każdego piksela rysowanego obiektu wyznacza się głębokość i porównuje ją z wartością zapamiętaną w buforze głębokości. Rysowanie i zapamiętanie głębokości piksela w buforze następuje tylko wtedy, gdy ta głębokość jest mniejsza.

199

Algorytm z buforem głębokości może posłużyć do wykonania końcowego obrazu, w którym piksele mają kolory obliczone na podstawie przyjętego modelu oświetlenia itd., lub może to być obliczenie pomocnicze, którego wyniki będą danymi dla kolejnych etapów wykonywania obrazu.

- Narysowanie sceny z punktu położenia źródła światła daje reprezentację **obszaru cienia**, co umożliwia otrzymanie obrazu z cieniami. W tym przypadku potrzebna bywa tylko końcowa zawartość bufora głębokości.
- Można narysować (poza ekranem) scenę widzianą z punktu położenia obserwatora odbitego w lustrze, a następnie użyć tego obrazu jako tekstury do nałożenia na lustro na końcowym obrazie.

200

- Zamiast ostatecznych kolorów w pikselach obrazu można zapamiętać inne atrybuty fragmentów zrasteryzowanych odcinków i wielokątów, na przykład numer (identyfikator) prymitywu, wektor normalny, współrzędne tekstury itp. Szader fragmentów nie musi w tym przypadku wykonywać skomplikowanych obliczeń koloru, który może później zostać zamalowany. Te obliczenia mogą być wykonane w kolejnym przebiegu rysowania, już tylko dla punktów, o których wiadomo, że są widoczne. Nie trzeba też wtedy powtarzać obliczeń związanych np. z rozdrabnianiem płatów.

Zbiór tablic z wymienionymi wyżej informacjami używanymi do późniejszego wykonywania końcowego obrazu jest nazywany **G-buforem**.

- Mając taki obraz, jak opisany wyżej, można użyć technik przetwarzania obrazu w celu wykonania obliczeń biorących pod uwagę sąsiedztwo piksela na obrazie. Na przykład na obrazie pomieszczenia oświetlenie wnek (np. wspólnych krawędzi ścian) bywa słabsze niż miejsc dalej od krawędzi. Inny możliwy efekt to otrzymanie poświaty wokół płomienia świecy.

201

- W sprzętowej realizacji algorytmu **śledzenia promieni** algorytm z buforem głębokości może posłużyć do znalezienia przecięć tzw. **promieni pierwotnych** (wychodzących z oka obserwatora) z obiektami w scenie.
- W **metodzie bilansu energetycznego** wykonuje się wiele obrazów sceny widzianej z punktów na powierzchniach obiektów w celu obliczenia tzw. **współczynnika kształtu** (*form factors*), które są współczynnikiem w układzie równań liniowych opisującym globalny rozkład oświetlenia w scenie (w której światło podlega wielokrotnym odbiciom od powierzchni obiektów).

202

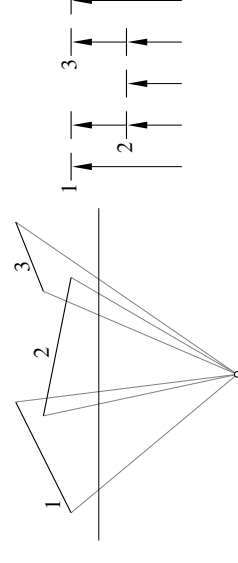
Liczne warianty **algorytmu przeglądania liniami poziomymi** mają obecnie głównie znaczenie historyczne. Algorytm ten był popularny, gdy dostępność pamięci RAM była bardzo ograniczona: z-bufor dla obrazu 640×480 z 16 bitami na piksel zajmuje 600kB, podczas gdy cała pamięć RAM w komputerach osobistych to było 640kB.

Algorytm polega na tym, że wszystkie wielokąty w scenie są jednocześnie poddawane rasteryzacji, linia po linii, przy czym w danej chwili są dostępne odcinki tych wielokątów odwzorowane na kolejną poziomą linię rastra.

Dzięki temu można utworzyć bufor głębokości dla tylko jednej linii rastra, zamiast dla wszystkich naraz.

203

Inna możliwość to posortowanie odcinków w kolejności rosnących współrzędnych x lewych końców i utworzenie listy aktywnych odcinków, posortowanej w kolejności rosnącej odległości od obserwatora. Ideą algorytmu przedstawia rysunek.



204

Algorytm umożliwia otrzymanie obrazu z cieniami. Dla każdego rysowanego odcinka można znaleźć jego część oświetloną i zaciemnioną, wyznaczając przekrój obiektów sceny z płaszczyzną zawierającą ten odcinek (w przestrzeni) i położenie źródła światła. Przekroje brył wielościennych są wielokątami, przekroje płaskich ścian są odcinkami. Można zatem zbadać widoczność tego odcinka z punktu położenia źródła światła i narysować go odpowiednio.

Dруга możliwość to dopuszczenie reprezentacji sceny w postaci drzewa CSG — z zatem można narysować scenę z takimi bryłami bez jawnego wyznaczania reprezentacji brzegowej bryły. Przecięcie bryły z płaszczyzną (wyznaczoną przez poziomą linię rastra i położenie obserwatora) jest wielokątem. Zadanie wyznaczania brzegu bryły CSG zostaje sprowadzone do zaania konstrukcyjnej geometrii płaskich wielokątów, czyli następuje redukcja wymiaru — rzecz zawsze opłacalna, bo radykalnie upraszczająca zadanie.

205

Algorytmy cieni

Mając punktowe źródła światła, możemy chcieć dla każdego punktu rysowanej powierzchni uwzględnić informację, które z tych źródeł bezpośrednio oświetlają ten punkt.

Dwa algorytmy cieni przestrzeni danych zasługują na wzmiankę: **algorytm Weitera–Athertona**, badający widoczność ścian (i ich części) z punktu położenia źródła światła i algorytm drzew BSP, budujący reprezentację **bryły cienia**.

Algorytmy przestrzeni obrazu to **algorytm Crowa** z 1977 r. (znany też jako **algorytm Carmacka**, użyty w grze *Doom 3*, 2000 r.), polegający na rysowaniu ścian bryły cienia i **algorytm z buforem głębokości**. Obecnie najmodniejszy jest chyba ten ostatni.

206

W **algorytmie brył cienia** scena jest rysowana dwukrotnie, przy czym za pierwszym razem potrzebne jest tylko wypełnienie bufora głębokości.

Dруги krok polega na narysowaniu ścian **elementarnych brył cienia**. Ściany te są ścianami sceny oraz czworokątami, których jedna krawędź jest krawędzią ściany sceny, dwie krawędzie są odcinkami prostych, na których leży źródło światła i odcinek krawędzi ściany, a czwarta krawędź jest dostatecznie daleko. Do tego dochodzi ściana zamykająca elementarną bryłę cienia, która jest ściętym ostrosłupem.

Wszystkie ściany elementarnej bryły cienia mają odpowiednią orientację: ich wektory normalne są skierowane na zewnątrz bryły. Rysowanie brył cienia odbywa się przy zablokowanym pisaniu do bufora głębokości, natomiast dla każdego piksela jest używany licznik, o początkowej wartości 0. Dla ścian odwróconych przodem do obserwatora licznik jest zwiększany, a dla ścian odwróconych tyłem zmniejszany o 1.

207

Po narysowaniu wszystkich ścian elementarnych brył cienia licznik dla danego piksela określa, ile razy więcej, przebywając odcinek od położenia obserwatora do widocznego punktu, weszliśmy do obszaru cienia niż z niego wyszliśmy. Jeśli licznik ma wartość 0, to przyjmujemy, że ten punkt jest oświetlony. Ten wariant algorytmu nazywa się *depth pass*.

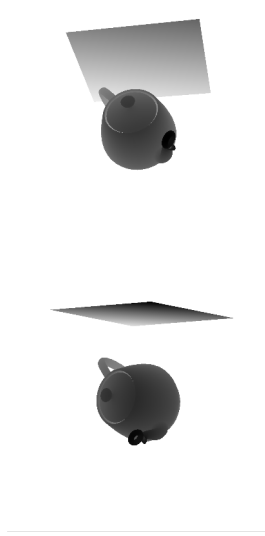
Wariant *depth fail* polega na odwróceniu testów widoczności podczas rysowania ścian brył cienia — licznik zmienia wartość tylko wtedy, gdy fragment jest niewidoczny, tj. ma *większą* głębokość niż ta zapamiętana w z-buforze. Przewagą tego wariantu jest to, że działa on poprawnie także wtedy, gdy sam obserwator znajduje się wewnątrz obszaru cienia.

Po narysowaniu brył cienia scenę rysuje się ponownie, z użyciem dostępnej informacji o oświetleniu dla każdego piksela.

Liczniki używane w tym algorytmie są zazwyczaj przechowywane w tzw. **buforze maski** (*stencil buffer*), dostępnym w OpenGL-u. Inne (i częstsze) jego zastosowanie polega na zdefiniowaniu obszaru zabronionego do rysowania.

208

W algorytmie z buforem głębokości scenę rysuje się w takim rzucie, aby rozstrzygnąć widoczność z punktu położenia źródła światła. Otrzymana przy tym zawartość bufora głębokości jest (przybliżoną) reprezentacją obszaru cienia, która podczas wykonywania końcowego obrazu służy do badania, czy fragment powierzchni jest oświetlony.



Pełną implementację tego algorytmu można znaleźć w moim skrypcie z OpenGL-a.

209

Modele oświetlenia

Na wygląd obiektów trójwymiarowych zasadniczy wpływ ma oświetlenie; kolor punktu na obrazie obiektu zależy od własności obiektu (zdolności jego powierzchni do odbijania światła, także emisja światła przez ten obiekt), własności źródła światła (położenia, mocy i widma promieniowania, kierunkowego rozkładu oświetlenia) oraz obserwatora (kierunku, z którego widzi on dany punkt).

Modele oświetlenia można podzielić na **empiryczne** (kolor punktu obliczamy według wzoru dobranego tak, aby wyszedł „ładny obrazek”) i **fizyczne**, w których wzór opisuje fizyczne cechy światła i jego oddziaływanie z odbijającą je powierzchnią.

Drugi podział to modele **lokalne** (bierzemy pod uwagę tylko własności światła i powierzchni w otoczeniu oświetlonego punktu) i **globalne**, w których bierzemy pod uwagę wielokrotne odbicia światła od obiektów w całej scenie.

210

Lokalne modele empiryczne

W modelach empirycznych wszystkie wielkości fotometryczne możemy wrzucić do jednego worka o nazwie „intensywność” światła, w szczególności padającego na powierzchnię i odbijającego się od niej.

Model Lamberta, wynaleziony przez J.H. Lamberta w 1760 r., zakłada, że obiekty są nieprzezroczyste i idealnie matowe. Niech \mathbf{n} oznacza jednostkowy wektor normalny powierzchni w danym punkcie; płaszczyzna prostopadła do wektora \mathbf{n} (czyli styczna do powierzchni) dzieli przestrzeń na dwie półprzestrzenie. Jeśli wektory \mathbf{l} , o kierunku do źródła światła, i \mathbf{v} , o kierunku do obserwatora, są w tej samej półprzestrzeni, to intensywność światła odbitego o ustalonej długości fali jest proporcjonalna do kosinusa kąta między wektorami \mathbf{l} a \mathbf{n} .

211

Często przyjmuje się, że źródło światła jest punktowe, co jednoznacznie określa wektor \mathbf{l} . Jeśli źródło jest położone (nieskończenie) daleko od sceny, to wektor \mathbf{l} oraz intensywność światła padającego na obiekty są stałe w całej scenie. Jeśli odległość źródła światła od obiektów jest skończona, to najczęściej przyjmuje się, że

$$I^{\text{dir}} = \frac{I^{\text{em}}}{ad^2 + bd + c}$$

gdzie I^{em} oznacza intensywność (moc) światła emitowanego przez źródło znajdujące się w odległości d od oświetlanego punktu, a I^{dir} jest to intensywność kierunkowego oświetlenia tego punktu. Współczynniki $a > 0$, $b \geq 0$, $c > 0$ doбира się empirycznie.

Intensywności I^{em} oraz I^{dir} są wielkościami wektorowymi, w istocie to są funkcje długości fali świetlnej, ale często przyjmuje się, że to są wektory o trzech współrzędnych, R , G , B , reprezentujących kolor światła wystarczająco dokładnie dla większości zastosowań w grafice.

212

Część światła z danego źródła rozprasza się między obiektami i oświetla powierzchnię z różnych kierunków. W najprostszym przypadku zakłada się, że intensywność tego światła nie zależy od kierunku. Po zsumowaniu intensywności odbitego światła pochodzącego ze wszystkich źródeł punktowych i uwzględnieniu światła rozproszonego w otoczeniu otrzymujemy wzór

$$L = a \sum_{i=0}^{n-1} (I_i^{\text{amb}} + I_i^{\text{dir}} v_i(|I_i, \mathbf{n}|)).$$

Czynnik v_i jest jedynką, jeśli obserwator widzi punkt na oświetlonej stronie powierzchni i zerem w przeciwnym przypadku. Wektory I_i są jednostkowe.

Wektor a opisuje zdolność powierzchni do odbijania światła dla poszczególnych długości fali, przy czym znów jest to najczęściej wektor o trzech współrzędnych, R , G , B . Mnożenie sumy przez ten wektor odbywa się „po współrzędnych”.

213

Model Phong, wynaleziony przez Bui-Tuong Phonga w 1975 r., pozwala narysować odbłaski na powierzchni niecałkowicie matowej. Wzór jest taki:

$$L = a \sum_{i=0}^{n-1} (I_i^{\text{amb}} + I_i^{\text{dir}} v_i(|I_i, \mathbf{n}|)) + s \sum_{i=0}^{n-1} I_i^{\text{dir}} W(\mathbf{n}, I_i, \mathbf{v}) \max\{0, (\mathbf{r}_i, \mathbf{v})\}^m.$$

Występuje w nim wektor jednostkowy

$$\mathbf{r}_i = 2(\mathbf{n}, I_i)\mathbf{n} - I_i,$$

który opisuje kierunek, w jakim foton padający z kierunku wektora I_i odbiłby się od idealnego lustra, którego wektorem normalnym jest wektor \mathbf{n} .

Za funkcję W bywa przyjmowana funkcja stała, równa 1, ale bywa też przyjmowana funkcja zależna od kątów między wektorami I_i i \mathbf{v} a wektorem \mathbf{n} . Dobiera się ją tak, aby zbliżyć efekty otrzymywane przy użyciu tego modelu do skutków użycia modeli fizycznych, a dokładniej do uwzględnienia mikrogeometrii (chropowatości) powierzchni i tzw. czynnika Fresnela. Wektor s zazwyczaj reprezentuje funkcję stałą (ma współrzędne $R \approx G \approx B$), bo kolor odbłasku jest zwykle kolorem światła padającego na powierzchnię.

214

Część światła z danego źródła rozprasza się między obiektami i oświetla powierzchnię z różnych kierunków. W najprostszym przypadku zakłada się, że intensywność tego światła nie zależy od kierunku. Po zsumowaniu intensywności odbitego światła pochodzącego ze wszystkich źródeł punktowych i uwzględnieniu światła rozproszonego w otoczeniu otrzymujemy wzór

$$L = a \sum_{i=0}^{n-1} (I_i^{\text{amb}} + I_i^{\text{dir}} v_i(|I_i, \mathbf{n}|)).$$

Czynnik v_i jest jedynką, jeśli obserwator widzi punkt na oświetlonej stronie powierzchni i zerem w przeciwnym przypadku. Wektory I_i są jednostkowe.

Wektor a opisuje zdolność powierzchni do odbijania światła dla poszczególnych długości fali, przy czym znów jest to najczęściej wektor o trzech współrzędnych, R , G , B . Mnożenie sumy przez ten wektor odbywa się „po współrzędnych”.

213

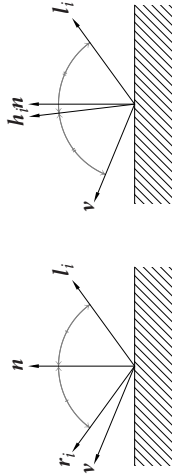
Model Blinna-Phonga powstał w roku 1977 jako modyfikacja modelu Phong.

Model ten jest opisany wzorem

$$L = a \sum_{i=0}^{n-1} (I_i^{\text{amb}} + I_i^{\text{dir}} v_i(|I_i, \mathbf{n}|)) + s \sum_{i=0}^{n-1} I_i^{\text{dir}} W(\mathbf{n}, I_i, \mathbf{v}) \langle \mathbf{h}_i, \mathbf{n} \rangle^{2m}.$$

Występuje w nim wektor

$$\mathbf{h}_i = \frac{1}{\|I_i + \mathbf{v}\|} (I_i + \mathbf{v}).$$



Wykładnik m w obu modelach odpowiada za „stopień wypolerowania” powierzchni — im jest większy, tym mniejszy jest obszar odbłasku.

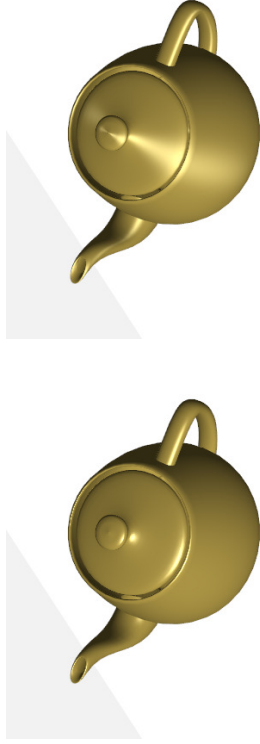
215

Modele Phong i Blinna-Phonga są **izotropowe**, powierzchnia nie ma w nich wyróżnionego kierunku. Często mamy do czynienia z powierzchniami, których chropowatości mają wyróżniony kierunek, np. są na nich rysy, niewidoczne gołym okiem, ale zmieniające kształt odbłasków. Najprostszym **model anizotropowy** jest modyfikacją modelu Blinna-Phonga, w nim wektory \mathbf{h}_i są zastąpione przez wektory $\hat{\mathbf{h}}_i$, takie że

$$\hat{\mathbf{h}}_i = \frac{1}{\|\hat{\mathbf{h}}_i\|} \hat{\mathbf{h}}_i, \quad \hat{\mathbf{h}}_i = P_0 \mathbf{w}_i + P_1 \mathbf{w}_i + a P_2 \mathbf{w}_i, \quad \mathbf{w}_i = I_i + \mathbf{v},$$

gdzie P_0, P_1, P_2 są to rzuty prostopadłe odpowiednio na kierunek wektora \mathbf{n} , kierunek pewnego wektora \mathbf{r} stycznego do powierzchni (wyznaczającego kierunek rys) i kierunek prostopadły do wektorów \mathbf{n} i \mathbf{r} . Współczynnik $a \in [0, 1]$ określa stopień anizotropii — dla $a = 1$ mamy $\hat{\mathbf{h}}_i = \mathbf{h}_i$, czyli model Blinna-Phonga, a wraz ze zmniejszaniem a rysy stają się coraz wyraźniejsze (choć cały czas ich nie widać).

216



$$a = 1$$

$$a = 0.2$$

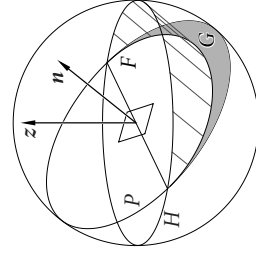
217

Jeśli przyjmiemy, że światło rozproszone w otoczeniu jest całkowicie bezkierunkowe, to efekt oświetlenia obiektów przy użyciu dowolnego modelu z dotychczas opisanych jest całkowicie nieplastyczny dla fragmentów powierzchni nieoświetlonych bezpośrednio. W związku z tym często przyjmuje się, że intensywność światła rozproszonego zależy od kierunku, z którego ono dochodzi i zamiast pierwszej sumy w podanych wzorach trzeba obliczyć pewną całkę.

Najprostszym jest **model oświetlenia hemisferycznego**. Zakładamy w nim, że światło dochodzi ze wszystkich kierunków. Wektory jednostkowe reprezentujące te kierunki tworzą sferę jednostkową, którą podzielimy na dwie półsfery: „górną” i „dolną”. Z kierunków górnej półsfery ochodzi światło jaśniejsze, o kolorze nieba, tj. białe lub niebieskawe. Z kierunków dolnej półsfery dochodzi światło słabsze, o kolorze podłoża (np. gruntu, trawy, skał, betonu, podłogi itp.).

Sferę podzielimy też na dwie półsfery płaszczyzną styczną do powierzchni. Zależnie od swojego położenia, obserwator widzi odbite światło, które doszło do powierzchni z kierunków jednej z tych półsfery.

218



Na rysunku mamy podziały sfery: okręgiem H , czyli „horyzontem” położonym w płaszczyźnie prostopadłej do wektora „zenitu” z i okręgiem P prostopadłym do wektora normalnego powierzchni, n . Mając dane intensywności światła f_{sky} i f_{ground} dochodzących od góry i od dołu (zakładamy, że w tych półsferach są stałe), chcemy obliczyć całkowitą intensywność światła odbitego w stronę obserwatora — przez zmieszanie w odpowiednich proporcjach tych dwóch światel.

219

Przyjmujemy, że to światło odbija się tylko „po lambertowski”, czyli tak, jak w powierzchni idealnie matowej. Wybierzmy pewien wektor I , należący do tej półsfery o brzegu P , w której jest wektor v kierunku do obserwatora i rozważmy mały fragment tej półsfery zawierający wektor I .

Intensywność światła dochodzącego z wszystkich (prawie identycznych) kierunków reprezentowanych przez ten fragment jest stała, a jego wkład w światło odbite przez powierzchnię jest proporcjonalny do kosinusa kąta padania, tj. iloczynu skalarnego wektorów I i n . Intensywność tę trzeba pomnożyć przez pole fragmentu półsfery i przez ten kosinus. Ale iloczyn pola fragmentu i kosinusa jest polem rzutu prostopadłego fragmentu na płaszczyznę styczną do powierzchni.

Rzuty wszystkich fragmentów przecięcia górnej półsfery z półsferą obserwatora wypełniają figurę F na rysunku — jest to suma połowy koła i połowy obszaru ograniczonego przez elipsę będącą rzutem połowy horyzontu (okręgu H) na płaszczyznę styczną.

220

Rzuty fragmentów przecięcia dolnej półsfery z półsferą obserwatora wypełniają figurę G na rysunku. Zatem, proporcja, w jakiej światła dochodzące z górnej i dolnej półsfery odbijają się od matowej powierzchni jest proporcją pól figur F i G — intensywność światła odbitego w modelu hemisferycznym jest równa

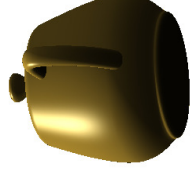
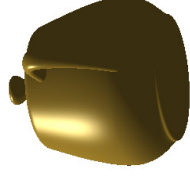
$$a((1-t)^{fsky} + tI^{ground}).$$

Jeśli kąt między wektorami z a n oznaczamy symbolem ϑ , to możemy (tzn. szader może) obliczyć

$$t = \begin{cases} (1 - \cos \vartheta)/2 = (1 - \langle z, n \rangle)/2 & \text{jeśli } \langle v, n \rangle > 0, \\ (1 + \cos \vartheta)/2 = (1 + \langle z, n \rangle)/2 & \text{jeśli } \langle v, n \rangle \leq 0. \end{cases}$$

221

Dodanie składnika opisującego lambertowskie oświetlenie hemisferyczne często daje też dobry efekt dla powierzchni błyszczących, a w każdym razie poprawia obraz.



Model oświetlenia przez światło dochodzące z wielu kierunków może być rozbudowany przez wprowadzenie bardziej skomplikowanej funkcji, opisującej światło dochodzące do obiektu otoczonego przez różne inne obiekty — na przykład budynki, ściany pomieszczenia itd. Obraz otoczenia obiektu służy do wyznaczenia tekstury, która opisuje światło odbite przez powierzchnię o wektorze normalnym n .

222

Rzuty fragmentów przecięcia dolnej półsfery z półsferą obserwatora wypełniają figurę G na rysunku. Zatem, proporcja, w jakiej światła dochodzące z górnej i dolnej półsfery odbijają się od matowej powierzchni jest proporcją pól figur F i G — intensywność światła odbitego w modelu hemisferycznym jest równa

$$a((1-t)^{fsky} + tI^{ground}).$$

Jeśli kąt między wektorami z a n oznaczamy symbolem ϑ , to możemy (tzn. szader może) obliczyć

$$t = \begin{cases} (1 - \cos \vartheta)/2 = (1 - \langle z, n \rangle)/2 & \text{jeśli } \langle v, n \rangle > 0, \\ (1 + \cos \vartheta)/2 = (1 + \langle z, n \rangle)/2 & \text{jeśli } \langle v, n \rangle \leq 0. \end{cases}$$

221

Są dwa podejścia do ilościowego opisywania światła:

- **Radiometria** — mierzymy moc światła (w watach) i wielkości pochodne, np. gęstość mocy na jednostkę powierzchni.
- **Fotometria** — mierzymy jasność (w lumenach), odpowiadającą subiektywnemu postrzeganiu światła przez ludzi. Światła o tej samej mocy, ale o różnych długościach fali, mają dla ludzi różne jasności.

Za zamianę mocy światła na subiektywne wrażenia odpowiada zmysł wzroku. Monitor komputera ma emitować światło o określonej mocy, dlatego radiometria odgrywa większą rolę w tworzeniu obrazów, czyli w grafice komputerowej.

224

Modele fizyczne

Rozchodzenie się światła w przestrzeni opisuje **elektrodynamika kwantowa**, która jest teorią zbyt skomplikowaną, aby dało się jej używać w grafice.

Jej uproszczeniem jest **optyka geometryczna**: fotony w jednorodnym ośrodku poruszają się po odcinkach prostych. Dalej, **optyka liniowa** zakłada, że fotony, jeśli nie są pochłaniane przez ośrodek, to zachowują swoją energię.

Te uproszczenia pomijają zjawiska dyfrakcji, interferencji i luminescencji.

223

Strumień świetlny (*flux*) mierzymy w watach [W]. Zakładając, że ośrodek, w którym rozchodzi się światło (np. powietrze) jest całkowicie przezroczysty, wywnioskujemy, że cała moc (czyli strumień) światła wpadającego do pewnego obszaru przestrzeni rozłoży się na tej części brzoju tego obszaru, przez którą światło z niego wychodzi.

Na początek rozważmy punktowe źródło światła monochromatycznego. Strumień świetlny przechodzący przez wszystkie sfery, których środek jest położeniem źródła światła, jest identyczny.

Intensywność kątową (*radiant intensity*) mierzymy w watach na steradian [W/sr].

Dla wycinka sfery *jednostkowej* jest to strumień światła wyemitowanego ze środka i przechodzącego przez ten wycinek, podzielonego przez pole wycinka.

225

Jeśli rozważamy wycinek sfery o dowolnym promieniu r , to strumień trzeba podzielić przez pole wycinka i pomnożyć przez r^2 . Pole wycinka podzielone przez kwadrat promienia jest miarą kąta bryłowego tego wycinka, mierzonego w steradianach [sr]. Pełny kąt bryłowy ma miarę 4π sr.

Punktowe źródło może promieniować inaczej w różnych kierunkach, a zatem intensywność kątowna jest w ogólności funkcją kierunku, z jakiego to źródło jest obserwowane. Ale w ośrodku całkowicie przezroczystym strumień światła przechodzącego przez dowolny wycinek sfery jednostkowej jest taki sam, jak przez otrzymany z niego za pomocą jednokładności o dodatnim współczynniku i tym samym ośrodku wycinek sfery o dowolnym promieniu.

226

Miara skróconego elementu jest to iloczyn $A |\cos \alpha(\mathbf{n}, \mathbf{v})|$ [m²]. Zatem kąt bryłowy elementu A w polu widzenia obserwatora jest ilorzem miary skróconego elementu i kwadratu jego odległości od obserwatora.

Teraz zajmijmy się źródłem światła, które nie jest punktem, tylko małym elementem powierzchni. Założymy, że jest to element płaski, o jednostkowym wektorze normalnym \mathbf{n} . Sam element, jak i jego pole, oznaczmy symbolem A .

Wektor \mathbf{v} wyznacza kierunek od punktu elementu A do obserwatora znajdującego się w odległości r — dużej w porównaniu ze średnicą elementu. W polu widzenia obserwatora element A ma kąt bryłowy $A |\cos \alpha(\mathbf{n}, \mathbf{v})|/r^2$; jeśli wektory \mathbf{n} i \mathbf{v} mają ten sam kierunek, to kąt bryłowy jest równy A/r^2 , a jeśli są wzajemnie prostopadłe, to kąt bryłowy jest równy 0.

Miara skróconego elementu jest to iloczyn $A |\cos \alpha(\mathbf{n}, \mathbf{v})|$ [m²]. Zatem kąt bryłowy elementu A w polu widzenia obserwatora jest ilorzem miary skróconego elementu i kwadratu jego odległości od obserwatora.

Wektor \mathbf{v} wyznacza kierunek od punktu elementu A do obserwatora znajdującego się w odległości r — dużej w porównaniu ze średnicą elementu. W polu widzenia obserwatora element A ma kąt bryłowy $A |\cos \alpha(\mathbf{n}, \mathbf{v})|/r^2$; jeśli wektory \mathbf{n} i \mathbf{v} mają ten sam kierunek, to kąt bryłowy jest równy A/r^2 , a jeśli są wzajemnie prostopadłe, to kąt bryłowy jest równy 0.

227

Radiancja (*radiance*) światła opuszczającego element (odbitego lub wypromieniowanego) jest równa strumieniowi świetlnemu podzielonemu przez miarę kąta bryłowego, w którym światło to zostało wysłane i miary skróconego elementu, przy założeniu, że rozpatrujemy kierunki przepływu światła bliskie kierunku pewnego wektora \mathbf{v} . Ze średnicą elementu A i z kierunkami przechodzimy do granic 0 i \mathbf{v} . W ustalonym punkcie powierzchni radiancja jest więc funkcją wektora jednostkowego \mathbf{v} , mierzoną w watach na metr kwadratowy i steradian [W/(m² sr)].

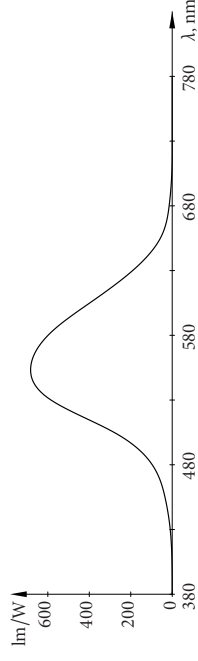
Irradiancja (*irradiance*) jest to gęstość mocy światła na jednostkę (nieskróconej) powierzchni [W/m²]. Otrzymujemy ją zatem dzieląc strumień świetlny przez pole elementu, przy czym interesuje nas najbardziej irradiancja światła padającego na element.

228

Wielkościom radiometrycznym odpowiadają wielkości fotometryczne:

- Strumień energetyczny [W] — **strumień świetlny** [lm].
- Intensywność kątowna [W/sr] — **światłość** [cd = lm/sr].
- Radiancja [W/(m²·sr)] — **luminancja** [lm/(m²·sr) = cd/m²].
- Irradiancja [W/m²] — **iluminancja** [lm/m²].

229



Funkcja zwana **skutecznością świetlną** (*luminous efficacy*) opisuje relację między tymi wielkościami w zależności od długości fali świetlnej λ . Maksymalną wartość 683 lm/W przyjmuje dla $\lambda = 555$ nm.

230

Subiektywne wrażenie jasności powierzchni zależy od radiancji L opuszczającego ją światła. Oczywiście, jest ono zależne od irradiancji obszaru siatkówki w oku, na którym powstaje obraz tej powierzchni.

Rozważmy element o polu A obserwowany z odległości r . Element ten zajmuje w polu widzenia obserwatora kąt $A|\cos \alpha$ (\mathbf{n}, \mathbf{v})/ r^2 . Pole obrazu elementu na siatkówce w oku jest proporcjonalne do tego kąta, z jakąś stałą C (jeśli element jest wprost przed obserwatorem).

Z kolei, widziana z punktu na elemencie A źrenica obserwatora zajmuje kąt bryłowy $\pi R^2/r^2$, gdzie R jest promieniem źrenicy, a więc strumień świetlny z elementu A wpadającego do oka jest iloczynem radiancji, miary skróconego elementu i tego kąta bryłowego. Dzieliąc strumień przez pole obrazu elementu A , otrzymamy irradiancję siatkówki:

$$I = (LA|\cos \alpha(\mathbf{n}, \mathbf{v})|\pi R^2/r^2)/(CA|\cos \alpha(\mathbf{n}, \mathbf{v})|/r^2) = \frac{\pi R^2}{C}L.$$

Zatem istotnie, jest ona proporcjonalna do radiancji L .

231

Dlaczego zatem inne gwiazdy nie są jasne jak Słońce? Bo ich ostre obrazy na siatkówce byłyby mniejsze niż receptory światła na siatkówce. Nawet gdyby obrazy gwiazd były całkowicie ostre, to strumień światła padający na receptor trzeba by podzielić przez pole jednego receptora, co daje mały iloraz.

Dwukierunkowa funkcja odbicia i załamania światła (*bidirectional scattering distribution function, BSDF*) jest własnością powierzchni — w danym punkcie jest ilorzazem radiancji światła odbitego od powierzchni lub załamane go w chwili przejścia przez nią i irradiancji światła padającego na ten punkt.

W ten sposób doszliśmy do sedna modeli fizycznych oświetlenia — model lokalny jest to w istocie dwukierunkowa funkcja odbicia i załamania. Bardziej szczegółowe modele uwzględniają jeszcze polaryzację światła i wtedy dwukierunkowa funkcja jest wektorowa.

232

Radiancja światła wysłanego z punktu \mathbf{p} elementu powierzchni A w kierunku wektora \mathbf{v} jest równa

$$L(\mathbf{p}, \mathbf{v}) = L_e(\mathbf{p}, \mathbf{v}) + L_r(\mathbf{p}, \mathbf{v}) = L_e(\mathbf{p}, \mathbf{v}) + \int_{I \in S} \rho(\mathbf{p}; I, \mathbf{v}) I(\mathbf{p}, I) dS.$$

L_e to radiancja światła emitowanego, L_r to radiancja światła odbitego, ρ oznacza dwukierunkową funkcję odbicia i załamania, a I irradiancję. S to sfera jednostkowa, tj. zbiór wszystkich kierunków, z których do punktu \mathbf{p} dochodzi światło.

To równanie w 1986 r. podał James T. Kajiya, nazywając je *The Rendering Equation*. Jeśli ośrodek (powietrze) jest całkownie przezroczysty, to wyraża ono radiancję punktu \mathbf{p} obserwowaną z kierunku \mathbf{v} , czyli to, co trzeba przedstawić na obrazie.

233

Modele fizyczne powinny spełniać dwie zasady:

Zasadę zachowania energii, zgodnie z którą strumień światła odbitego od powierzchni może być tylko mniejszy od strumienia światła padającego (część energii światła padającego rozprasza się w postaci ciepła), oraz

Zasadę Helmholtza, która bierze się z postulatu, że jeśli pewien foton, odbijając się od powierzchni, może przebyć pewną drogę, to inny foton może przebyć tę samą drogę w przeciwną stronę. Konsekwencją tej zasady jest symetria dwukierunkowej funkcji odbicia i załamania światła, ρ . W dowolnym punkcie \mathbf{p} , dla dowolnego wektora I kierunku padania światła i dowolnego wektora \mathbf{v} kierunku odbicia lub załamania światła zachodzi równość

$$\eta_1^2 \rho(\mathbf{p}; I, \mathbf{v}) = \eta_2^2 \rho(\mathbf{p}; \mathbf{v}, I),$$

gdzie η_1 i η_2 to współczynniki załamania światła ośrodków, w których foton porusza się przed i po oddziaływaniu z powierzchnią.

234

Strumień energetyczny światła padającego z kierunku ustalonego wektora I na element A jest równy

$$E = AI(\mathbf{p}, I).$$

Całkowita moc światła, które po dościsaniu z kierunku I zostało odbite lub załamane we wszystkich kierunkach, jest równa

$$\int_{\mathbf{v} \in S} L(\mathbf{p}, \mathbf{v}) A |\cos \angle(\mathbf{n}, \mathbf{v})| dS = AI(\mathbf{p}, I) \int_{\mathbf{v} \in S} \rho(\mathbf{p}; I, \mathbf{v}) |\cos \angle(\mathbf{n}, \mathbf{v})| dS.$$

Ponieważ ta moc jest mniejsza niż E (bo część energii światła zamienia się w ciepło), musi być

$$\int_{\mathbf{v} \in S} \rho(\mathbf{p}; I, \mathbf{v}) |\cos \angle(\mathbf{n}, \mathbf{v})| dS < 1$$

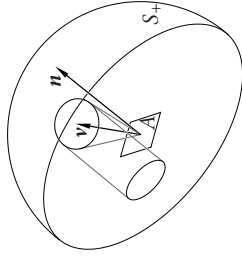
dla każdego wektora jednostkowego I .

235

W modelu Lamberta dwukierunkowa funkcja odbicia i załamania światła w danym punkcie przyjmuje dwie wartości: pewną stałą $\rho > 0$, jeśli iloczyn skalarny (I, \mathbf{n}) i (\mathbf{v}, \mathbf{n}) mają ten sam znak i 0 w przeciwnym przypadku. Zauważmy, że kosinus kąta między wektorami I a \mathbf{n} w tym modelu jest czynnikiem irradiancji światła padającego na powierzchnię, bo do niego jest proporcjonalny strumień światła. Łatwo jest sprawdzić, że model Lamberta spełnia zasadę Helmholtza.

Całkowita moc światła odbitego w modelu Lamberta może być obliczona po półsfery (dla $(\mathbf{n}, I) \cdot (\mathbf{n}, \mathbf{v}) > 0$, oznaczmy ją S_+), w której funkcja ρ jest dodatnia (ma stałą wartość $\rho(\mathbf{p}) > 0$), bo w drugiej półsfery jest zerowa. Możemy ją obliczyć bez rachunków.

236



$$\rho(\mathbf{p}) \int_{v \in S_+} |\cos \angle(\mathbf{n}, \mathbf{v})| dS = \rho(\mathbf{p})\pi.$$

Stąd, aby zasada zachowania energii była spełniona, musi być $\rho(\mathbf{p}) < \frac{1}{\pi}$.

237

Spełnienie zasady zachowania energii przez modele Phonga, Bui-Tuong Phonga i inne zależy od parametrów tych modeli, ale sprawdzenie, czy po ich dobraniu zasada jest spełniona, jest trudniejsze niż dla modelu Lamberta.

Model Phonga nie spełnia zasady Helmholtza, natomiast model Blinn-Phonga spełnia ją, jeśli występująca w nim funkcja W może być przedstawiona w postaci

$$W(\mathbf{n}, \mathbf{l}, \mathbf{v}) = |\langle \mathbf{n}, \mathbf{l} \rangle| Z(\mathbf{n}, \mathbf{l}, \mathbf{v}),$$

w której czynnik $Z(\mathbf{n}, \mathbf{l}, \mathbf{v})$ jest funkcją symetryczną: $Z(\mathbf{n}, \mathbf{l}, \mathbf{v}) = Z(\mathbf{n}, \mathbf{v}, \mathbf{l})$ dla wszystkich wektorów \mathbf{v} i \mathbf{l} .

238

Dwukierunkową funkcję odbicia i załamania zazwyczaj przedstawia się w postaci sumy:

$$\rho(\mathbf{p}; \mathbf{l}, \mathbf{v}) = \rho_r(\mathbf{p}; \mathbf{l}, \mathbf{v}) + \rho_t(\mathbf{p}; \mathbf{l}, \mathbf{v});$$

składnik ρ_r to **dwukierunkowa funkcja odbicia światła** (*bidirectional reflectance distribution function, BRDF*), a ρ_t to **dwukierunkowa funkcja przechodzenia światła** przez powierzchnię (*bidirectional transmission distribution function, BTDF*).

Jeśli powierzchnia jest nieprzezroczysta, to funkcja ρ_t jest zerowa.

Najczęściej używane modele fizyczne przyjmują funkcję ρ_r o postaci

$$\rho_r = v(k_d \rho_d + k_s \rho_s).$$

Czynnik v jest jedynką lub zerem, zależnie od tego, czy obserwator i źródło światła są po tej samej stronie płaszczyzny elementu, czy po przeciwnych stronach. Jest $k_d, k_s > 0$, $k_d + k_s = 1$. Pierwszy składnik w nawiasie odpowiada za odbicie rozproszone, a drugi składnik określa odbicie zwierciadlane.

239

Funkcja ρ_s w modelach fizycznych jest opisana wzorem

$$\rho_s = \frac{DGF\lambda}{4\langle \mathbf{l}, \mathbf{n} \rangle \langle \mathbf{v}, \mathbf{n} \rangle}.$$

Iloczyn skalarny wektorów jednostkowych w mianowniku to kosinusy kątów między tymi wektorami.

Zakłada się, że powierzchnia jest chropowata i składa się z mikroskopijnych ścianek, które są doskonałymi lustrami, przy czym każda prosta o kierunku wektora \mathbf{n} przecina tylko jedną mikrościankę.

Czynnik $D = D(\mathbf{h})$ opisuje rozkład kierunków wektorów normalnych mikrościanek. Dokładniej, określa on, jak wiele tych mikrościanek jest ustawionych tak, że ich jednostkowym wektorem normalnym jest wektor $\mathbf{h} = \frac{1}{\|\mathbf{l} + \mathbf{v}\|}(\mathbf{l} + \mathbf{v})$. Zawsze ma być

$$\int_{\mathbf{h} \in S_+} D(\mathbf{h}) \cos \angle(\mathbf{h}, \mathbf{n}) dS = 1.$$

Zbiór S_+ składa się z tych wektorów jednostkowych \mathbf{h} , dla których $\langle \mathbf{h}, \mathbf{n} \rangle > 0$.

240

W modelu **Torrancéa i Sparrowa** (1967 r.) został użyty rozkład Gaussa:
 $D(\vartheta) = ae^{-b^2\vartheta^2}$, gdzie ϑ oznacza kąt między wektorem normalnym mikrościanki a wektorem normalnym powierzchni (trzeba obciąć ϑ do przedziału $[0, \pi/2)$).
 Współczynniki a i b są stałymi. Po ustaleniu b trzeba dobrać a , co jest kłopotliwe.

241

W modelu **Cooka i Torrancéa** (1981 r.) jest przyjęty rozkład Beckmanna–Spizzichino,

$$D_{BS}(\mathbf{h}) = \frac{e^{-(\lg \vartheta)^2/m^2}}{\pi m^2 \cos^4 \vartheta}.$$

Parametr m określa chropowatość; zmniejszanie tego parametru powoduje większe „wypolerowanie” powierzchni. W obliczeniach można użyć wzoru

$$\frac{\lg^2 \vartheta}{m^2} = \frac{1 - \cos^2 \vartheta}{m^2 \cos^2 \vartheta},$$

oczywiście $\cos \vartheta = \langle \mathbf{n}, \mathbf{h} \rangle$.

Wersja anizotropowa — są dwa parametry chropowatości, m_u, m_v :

$$D_{BSa}(\mathbf{h}) = \frac{e^{-(c^2/m_u^2 + s^2/m_v^2) \lg^2 \vartheta}}{\pi m_u m_v \cos^4 \vartheta}.$$

We wzorze tym c i s to odpowiednio kosinus i sinus kąta między rzutem wektora \mathbf{h} na płaszczyznę styczną do powierzchni a ustalonym wektorem \mathbf{u} w tej płaszczyźnie.

242

Często jest też używany np. w grach rozkład **Trowbridge’a–Reitza**:

$$D_{TR}(\mathbf{h}) = \frac{m^2}{\pi(1 + (m^2 - 1)^2 \cos^4 \vartheta)}.$$

Jego wersja anizotropowa jest opisana wzorem

$$D_{TRa}(\mathbf{h}) = \frac{1}{\pi m_u m_v (1 + \sin^2 \vartheta (c^2/m_u^2 + s^2/m_v^2))^2 \cos^4 \vartheta}.$$

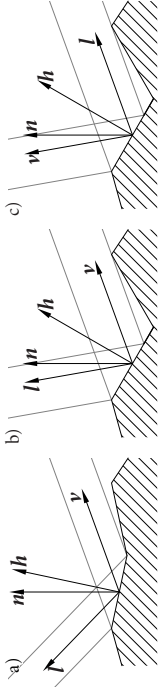
243

Istnieją powierzchnie, których mikrościanki należą do kilku podzbiorów, w których są rozłożone zgodnie z różnymi rozkładami, przy czym może to być ten sam rozkład (np. Beckmanna–Spizzichino) z różnymi parametrami (takimi jak m).

Powierzchnia taka mogła być np. poddana obróbce zgrubnej (szlifowaniu), a potem niezbyt dokładnie wypolerowana.

Aby otrzymać taki efekt na obrazie, ustala się wagi poszczególnych rozkładów (tj. dodatnie parametry sumujące się do 1) i przyjmuje funkcję D jako kombinację wypukłą poszczególnych rozkładów z tymi wagami.

244



Czynnik G opisuje stopień zasilania części mikrościanek przez inne elementy powierzchni; mikrościanka może nie być cała oświetlona przez światło padające z kierunku \mathbf{l} i może nie być cała widoczna z kierunku \mathbf{v} . Przyjmując hipotezę, że mikrościanki występują parami, tworząc rowki, takie że kąty między ich wektorami normalnymi a wektorem \mathbf{n} są jednakowe, można otrzymać wzór

$$G = \min \left\{ 1, \frac{2(\mathbf{h}, \mathbf{n}) \langle \mathbf{v}, \mathbf{n} \rangle}{\langle \mathbf{v}, \mathbf{h} \rangle}, \frac{2(\mathbf{h}, \mathbf{n}) \langle \mathbf{l}, \mathbf{n} \rangle}{\langle \mathbf{l}, \mathbf{h} \rangle} \right\}.$$

245

Wreszcie, czynnik Fresnela F_λ zależy od własności materiału, którego powierzchnia jest oświetlana, a ściślej od jego współczynnika załamania światła (który zależy od długości fali).

Dla światła niespolaryzowanego

$$F_\lambda = \frac{1}{2} \frac{(g-c)^2}{(g+c)^2} \left(1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2} \right),$$

w którym $c = \cos \vartheta_{HI} = \langle \mathbf{h}, \mathbf{l} \rangle$, $g = \sqrt{(\eta_{\lambda,2}/\eta_{\lambda,1})^2 + c^2} - 1$, a symbole $\eta_{\lambda,1}$ i $\eta_{\lambda,2}$ oznaczają współczynnik załamania światła ośrodka, przez który światło dochodzi do powierzchni obiektu i współczynnik załamania światła materiału, z którego jest wykonany ten obiekt.

246

W praktyce czynniki Fresnela są zastępowane wyrażeniami łatwiejszymi do obliczenia. Symbolami $F_{\lambda,0}$ i $F_{\lambda,\pi/2}$ oznaczmy czynniki Fresnela dla światła padającego na mikrościankę prostopadle oraz stycznie do niej. W pierwszym przypadku $\mathbf{l} = \mathbf{h}$, skąd wynika, że $c = 1$ i $g = \eta_{\lambda,2}/\eta_{\lambda,1}$, a w drugim mamy $c = 0$ i $g = \sqrt{(\eta_{\lambda,2}/\eta_{\lambda,1})^2 - 1}$. Stąd

$$F_{\lambda,0} = \left(\frac{\eta_{\lambda,2} - \eta_{\lambda,1}}{\eta_{\lambda,2} + \eta_{\lambda,1}} \right)^2, \quad F_{\lambda,\pi/2} = 1.$$

Przybliżenie Schlicka, opisane wzorem

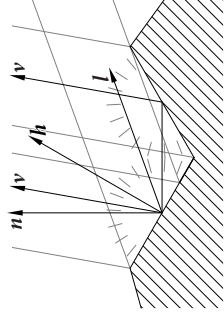
$$F_{\lambda,S} = F_{\lambda,0} + (F_{\lambda,\pi/2} - F_{\lambda,0})(1 - \cos \vartheta_{HI})^5 = F_{\lambda,0} + (1 - F_{\lambda,0})(1 - \langle \mathbf{h}, \mathbf{l} \rangle)^5,$$

wymaga podania tylko funkcji $F_{\lambda,0}$, której argumentem jest długość fali świetlnej λ , przy czym funkcja ta jest często reprezentowana przez trzy wartości otrzymane z pomiarów współczynnika załamania światła czerwonego, zielonego i niebieskiego.

247

Funkcja ρ_d , opisująca odbicie rozproszone, bardzo często jest przyjmowana taka, jak w modelu Lamberta, który zakłada idealną gładkość powierzchni. Chropowatość powierzchni uwzględnia model Orena i Nayara, który zakłada istnienie mikrościanek odbijających światło w sposób lambertowski — ale mikrościanki mają różne kierunki wektorów normalnych i mogą się wzajemnie zasłaniać.

Model zakłada, że pary mikrościanek tworzą symetryczne rowki; do obserwatora dochodzi światło odbite przez jedną lub obie mikrościanki z pary.



248

Rozkład kierunków wektorów normalnych mikrościanek jest gaussowski (z parametrem chropowatości σ). Podane przez twórców modelu wzory opisują całkę po rozkładzie w przybliżeniu:

$$\rho_{d,1} = \frac{c_1}{\pi} \left(C_1 + cC_2 \operatorname{tg} \beta + (1 - |c|) C_3 \operatorname{tg} \frac{\alpha + \beta}{2} \right).$$

Występujące w nim symbole oznaczają

$$\begin{aligned} \alpha &= \max\{\vartheta_I, \vartheta_V\}, & \beta &= \min\{\vartheta_I, \vartheta_V\}, \\ C_1 &= 1 - \frac{0.5\sigma^2}{\sigma^2 + 0.33}, & C_2 &= \frac{0.45\sigma^2}{\sigma^2 + 0.09}d, & C_3 &= \frac{0.125\sigma^2}{\sigma^2 + 0.09} \left(\frac{4\alpha\beta}{\pi^2} \right)^2, \\ c &= \cos \Delta\varphi, & d &= \begin{cases} \sin \alpha & \text{jeśli } c \geq 0, \\ \sin \alpha - \left(\frac{2\beta}{\pi} \right)^3 & \text{w przeciwnym razie.} \end{cases} \end{aligned}$$

Funkcja c_1 o wartościach w przedziale $[0, 1]$ opisuje, jaki ułamek mocy światła o długości fali λ jest odbijany przez mikrościankę. Symbole ϑ_I i ϑ_V oznaczają kąty między wektorami \mathbf{l} i \mathbf{v} a wektorem \mathbf{n} : jest $\cos \vartheta_I = \langle \mathbf{l}, \mathbf{n} \rangle$ i $\cos \vartheta_V = \langle \mathbf{v}, \mathbf{n} \rangle$.

249

Kąt $\Delta\varphi$ jest mierzony między rzutami prostopadłymi wektorów \mathbf{l} i \mathbf{v} na płaszczyznę styczną do powierzchni. Jeśli oba te wektory nie mają kierunku wektora \mathbf{n} , to

$$c = \frac{\langle \mathbf{l}, \mathbf{v} \rangle - \langle \mathbf{n}, \mathbf{l} \rangle \langle \mathbf{n}, \mathbf{v} \rangle}{\| \mathbf{l} - \langle \mathbf{n}, \mathbf{l} \rangle \mathbf{n} \| \| \mathbf{v} - \langle \mathbf{n}, \mathbf{v} \rangle \mathbf{n} \|}.$$

W przeciwnym razie $\beta = \operatorname{tg} \beta = C_3 = 0$, $d = \sin \alpha$ i nieokreślona liczba c jest nieistotna.

250

Jeśli obiekt jest wykonany z materiału przezroczystego (np. szkła), ale jego powierzchnia jest chropowata, to składnikiem funkcji ρ jest **dwukierunkowa funkcja przechodzenia światła** (*bidirectional transmission distribution function*, BTDF), opisana wzorem

$$\rho_{\mathbf{l}} = -\frac{DG(1 - F_\lambda)}{4\langle \mathbf{l}, \mathbf{n} \rangle \langle \mathbf{v}, \mathbf{n} \rangle \eta_1^2} \eta_2^2, \quad (1)$$

w którym występują te same funkcje D , G i F_λ . Ponieważ czynnik Fresnela F_λ określa, jaka część fotonów odbija się od mikrościanki, czynnik $1 - F_\lambda$ opisuje fotony, które przez nią przechodzą. Czynniki D i G opisują rozkład kierunków wektorów normalnych mikrościanek i ich wzajemne zasłanianie, ale teraz obliczenie wektora normalnego mikrościanki \mathbf{h} dla danych wektorów \mathbf{l} i \mathbf{v} jest trochę bardziej skomplikowane.

252

Funkcja $\rho_{d,1}$ jest składnikiem funkcji ρ_d opisującym pojedyncze odbicie światła w kierunku \mathbf{v} od mikrościanek oświetlonych bezpośrednio. Uwzględnienie światła wysłanego w tym kierunku po dwóch kolejnych odbiciach (w obu mikrościankach tworzących każdą parę) wymaga dodania składnika

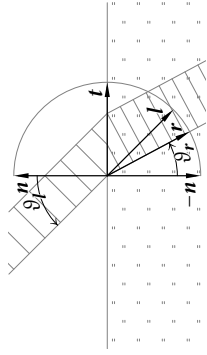
$$\rho_{d,2} = \frac{c_\lambda^2}{\pi} \frac{0.17\sigma^2}{\sigma^2 + 0.13} \left(1 - c \frac{4\beta^2}{\pi^2} \right).$$

W praktyce składnik ten bywa pomijany. Podstawiając $\sigma = 0$, otrzymamy $C_1 = 1$, $C_2 = C_3 = \rho_{d,2} = 0$, wskutek czego powstanie lambertowski model odbicia światła.

251

Foton poruszający się w kierunku wektora \mathbf{l} po przejściu rzez granicę ośrodków uda się w kierunku \mathbf{r} .

Uwaga: tu zmieniłem zwrot wektora \mathbf{l} , przyjmując konwencję taką, jak w GLSL-owej funkcji `refract`, która oblicza wektor \mathbf{r} .



Kąt między wektorami \mathbf{l} a \mathbf{n} oznaczmy ϑ_l , a kąt załamania światła ϑ_r .

253

Kąty ϑ_{lml} i ϑ_r są związane prawem Snella:

$$\eta_1 \sin \vartheta_l = \eta_2 \sin \vartheta_r,$$

w którym η_1 i η_2 to współczynniki załamania światła.

Oznaczmy $\eta = \eta_1/\eta_2$. Wtedy

$$\begin{aligned} \sin \vartheta_r &= \eta \sin \vartheta_l, \\ \cos \vartheta_r &= \sqrt{1 - \sin^2 \vartheta_r} = \sqrt{1 - \eta^2 \sin^2 \vartheta_l} = \sqrt{1 - \eta^2 (1 - \cos^2 \vartheta_l)}. \end{aligned}$$

Mamy $\cos \vartheta_l = -\langle \mathbf{n}, \mathbf{l} \rangle$. Niech $\mathbf{t} = \frac{1}{\sin \vartheta_l} (\mathbf{l} - \langle \mathbf{n}, \mathbf{l} \rangle \mathbf{n})$. Wektor \mathbf{t} jest jednostkowy i styczny do powierzchni. Za jego pomocą możemy obliczyć wektor jednostkowy

$$\begin{aligned} \mathbf{r} &= -\mathbf{n} \cos \vartheta_r + \mathbf{t} \sin \vartheta_r = \mathbf{n} \cos \vartheta_r + \mathbf{t} \eta \sin \vartheta_l \\ &= \mathbf{n} \sqrt{1 - \eta^2 (1 - \langle \mathbf{n}, \mathbf{l} \rangle^2)} + \eta (\mathbf{l} - \langle \mathbf{n}, \mathbf{l} \rangle \mathbf{n}) \\ &= \eta \mathbf{l} - (\sqrt{1 - \eta^2 (1 - \langle \mathbf{n}, \mathbf{l} \rangle^2)} + \eta \langle \mathbf{n}, \mathbf{l} \rangle) \mathbf{n}. \end{aligned}$$

254

Oznaczając $k = 1 - \eta^2 (1 - \langle \mathbf{n}, \mathbf{l} \rangle^2)$, dostaniemy

$$\mathbf{r} = \eta \mathbf{l} - (\sqrt{k} + \eta \langle \mathbf{n}, \mathbf{l} \rangle) \mathbf{n}.$$

Jeśli $k \geq 0$, to $k = \cos^2 \vartheta_r$, ale może być $k < 0$ — w tym przypadku następuje całkowite odbicie wewnętrzne światła przebiegającego w gęstszym ośrodku optycznym.

Teraz w miejsce \mathbf{l} , \mathbf{r} i \mathbf{n} podstawimy wektory $-\mathbf{l}$, \mathbf{v} oraz \mathbf{h} — wektor normalny mikrościanki. Ze wzoru

$$\mathbf{v} = -\eta \mathbf{l} - (\sqrt{k} - \eta \langle \mathbf{h}, \mathbf{l} \rangle) \mathbf{h}$$

otrzymamy wektor jednostkowy

$$\mathbf{h} = \frac{\pm 1}{\|\mathbf{v} + \eta \mathbf{l}\|} (\mathbf{v} + \eta \mathbf{l}).$$

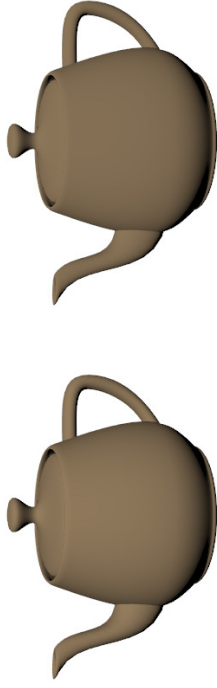
Znak wybieramy tak, aby było $\langle \mathbf{n}, \mathbf{h} \rangle > 0$. Na podstawie wektora \mathbf{h} możemy obliczyć wartość funkcji ρ_t .

255



Model Cooka i Torrance'a, czajnik stalowy i miedziany, jedno (punktowe) źródło światła.

256



Modele Lamberta i Orena–Nayara, czajnik z terakoty.

257

Całkowanie radiancji w modelu Lamberta

Radiancja światła odbitego przez powierzchnię nieprzezroczystego obiektu w modelu Lamberta jest opisana wzorem

$$L_r(\mathbf{p}) = \rho(\mathbf{p}) \int_{I \in S_+} L(\mathbf{p} + \mathbf{l}, -\mathbf{l}) \langle \mathbf{l}, \mathbf{n} \rangle dS = \rho(\mathbf{p}) I(\mathbf{p}, \mathbf{n}).$$

Funkcja $L(\mathbf{p} + \mathbf{l}, -\mathbf{l})$ opisuje radiancję światła dochodzącego ze wszystkich stron do punktu \mathbf{p} .

Jeśli obiekt jest mały w porównaniu z odległością od otaczających go przedmiotów, to można obliczyć irradiancję dla wszystkich kierunków wektora \mathbf{n} i dla jednego ustalonego punktu \mathbf{p} . Tak obliczoną irradiancję można (i warto) zastąpić (określona nieformalnie) intensywność światła idealnie rozproszonego w otoczeniu w *empirycznym* modelu oświetlenia Lamberta (lub np. Blinna–Phonga).

258

Do reprezentowania funkcji określonych na sferze służą w OpenGL-u tzw. **tekstury kostkowe**, które składają się z sześciu kwadratowych tablic teksteł. Argumentem funkcji **texture** w OpenGL-u jest niezerowy wektor $\mathbf{w} \in \mathbb{R}^3$, funkcja oblicza przecięcie półprostej $\{t\mathbf{w} : t > 0\}$ z jedną ze ścian sześciianu $[-1, 1]^3$ i dokonuje interpolacji teksteł.

Tekstura kostkowa opisująca tło dla rysowanego obiektu, np. podłogę, ściany, sufit i meble w pomieszczeniu lub krajobraz, tj. grunt, budynki, drzewa lub inne przedmioty dookoła i niebo, reprezentuje też radiancję. Na jej podstawie można utworzyć teksturę kostkową reprezentującą irradiancję — każdy texel jest wartością całki dla odpowiedniego wektora \mathbf{n} .

Ta technika nazywana jest **oświetleniem przez obraz** (*image-based lighting*, *IBL*).

259

Punktem wyjścia jest tekstura kostkowa opisująca oświetlenie ze wszystkich stron — to jest po prostu obraz otaczających przedmiotów, składający się na przykład z fotografii. Dla dowolnego wektora $\mathbf{l} \neq \mathbf{0}$ jej wartość jest radiancją światła padającego na punkt \mathbf{p} z kierunku \mathbf{l} .

Na jej podstawie trzeba skonstruować teksturę kostkową, której wartość dla każdego wektora $\mathbf{n} \neq \mathbf{0}$ jest irradiancją punktu \mathbf{p} powierzchni o wektorze normalnym \mathbf{n} . Irradiancję obliczymy dla wektorów \mathbf{n} odpowiadających texselom: rozdzielczość tej tekstury może być mała, wystarczy $6 \times 64 \times 64$.

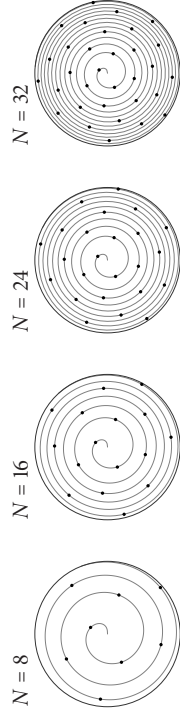


260

Dla każdego wektora \mathbf{n} trzeba obliczyć całkę po półsfery S_+ , która jest zbiorem wektorów jednostkowych \mathbf{l} , takich że $(\mathbf{l}, \mathbf{n}) \geq 0$. Funkcja podcałkowa jest iloczynem $L(\mathbf{p} + \mathbf{l}, -\mathbf{l})(\mathbf{l}, \mathbf{n})$. Dlatego można użyć kwadratury otrzymanej przez rozmieszczenie N punktów w kole jednostkowym z równomierną gęstością.

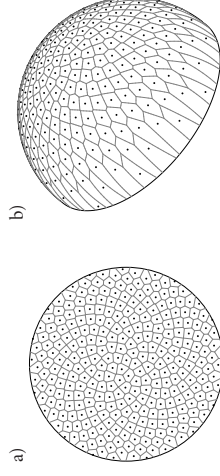
$$x_i = r_i \cos \varphi_i, \quad y_i = r_i \sin \varphi_i, \quad i = 0, \dots, N-1, \quad (2)$$

w których $r_i = \sqrt{(2i+1)/(2N)}$, $\varphi_i = (i + \frac{1}{2})2\pi\tau^2$, $\tau = (\sqrt{5}-1)/2$.



Aby otrzymać wektory \mathbf{l}_i , trzeba wektory $\tilde{\mathbf{l}}_i = (x_i, y_i, \sqrt{1-x_i^2-y_i^2})$ przekształcić za pomocą izometrii przeprowadzającej wektor $\mathbf{e}_3 = (0, 0, 1)$ na \mathbf{n} .

Izometria ta jest odbiciem Householdera lub złożeniem takiego odbicia i zmiany zwrotu na przeciwny.



Kwadratura jest określona wzorem

$$Q(L) = \frac{\pi}{N} \sum_{i=0}^{N-1} L(\mathbf{p} + \mathbf{l}_i, -\mathbf{l}_i).$$

Obliczenie może być wykonane przez GPU — należy narysować ściany sześcianu na teksturze kostkowej. Program zawiera szader fragmentów w GLSL-u, który oblicza wartość kwadratury:

```
#version 450 core
#define PI 3.141592653
#define DPHI 2.3999963229
in vec3 Normal;
layout(location=0) out vec4 out_Colour;
layout(binding=0) uniform samplerCube RadianceTxt;
uniform int N = 1500;
void main ( void )
{
    int i;
    float phi, r, gamma;
    vec3 l, w, irrad;
```

```
w = normalize ( Normal );
w.z += Normal.z > 0.0 ? 1.0 : -1.0;
gamma = 2.0 / dot ( w, w );
irrad = vec3(0.0);
for ( i = 1, phi = 0.5*DPHI; i < 2*N; i += 2, phi += DPHI ) {
    r = sqrt ( float(i)/float(2*N) );
    l = vec3 ( r*cos ( phi ), r*sin ( phi ),
              sqrt ( float(2*N-i)/float(2*N) ) ); /* sqrt ( 1 - r*r ) */
    l -= w*(gamma*dot ( w, l ));
    if ( Normal.z > 0.0 ) l = -l;
    irrad += texture ( RadianceTxt, l ).rgb;
}
out_Colour = vec4 ( irrad*(PI/float(N)), 1.0 );
} /*main*/
```

Znacznie mniej miejsca niż tekstura zajmuje przybliżający funkcję I wielomian drugiego stopnia zmiennych x, y, z , które są współrzędnymi wektora \mathbf{n} . Ten sposób zaproponowali Ramamoorthi i Hanrahan.

Dziedzina funkcji I , jak i przybliżającego ją wielomianu p jest sfera jednostkowa S . Przestrzeń wielomianów stopnia 2 trzech zmiennych ma wymiar 10, ale po obcięciu dziedziny do sfery S (o równaniu $x^2 + y^2 + z^2 - 1 = 0$) pozostaje przestrzeń o wymiarze 9, bo wielomiany różniące się o składnik podzielny przez wielomian $q(x, y, z) = x^2 + y^2 + z^2 - 1$ są na sferze S nierozróżnialne.

W zbiorze wielomianów, które resztę z dzielenia przez q mają taką samą, można znaleźć (dokładnie jeden) wielomian spełniający równanie Laplace'a, czyli funkcję harmoniczną.

265

Baza przestrzeni V , której elementami są wielomiany harmoniczne stopnia co najwyżej 2, składa się z wektorów

$$p_0 = 1, \quad p_1 = x, \quad p_2 = y, \quad p_3 = z, \quad p_4 = xy, \quad p_5 = yz, \quad p_6 = zx, \\ p_7 = x^2 - y^2, \quad p_8 = 2z^2 - x^2 - y^2.$$

To jest baza ortogonalna w sensie iloczynu skalarnego

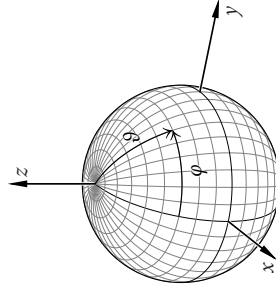
$$\langle f, g \rangle = \int_{w \in S} f(w)g(w) \, dS.$$

Korzystając z niej, można znaleźć wielomian p jako rzut ortogonalny funkcji I na przestrzeń V :

$$p = \sum_{i=0}^8 \frac{\langle p_i, I \rangle}{\|p_i\|^2} p_i,$$

trzeba tylko obliczyć współczynniki $a_i = \langle p_i, I \rangle / \|p_i\|^2$.

266



Sfera S ma parametryzację

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} (\varphi, \vartheta) = \begin{bmatrix} \cos \varphi \sin \vartheta \\ \sin \varphi \sin \vartheta \\ \cos \vartheta \end{bmatrix}, \quad \varphi \in [0, 2\pi], \quad \vartheta \in [0, \pi].$$

267

Potrzebne są całki po S z wielomianów $x^i y^j z^k$ dla $i + j + k \leq 4$. Miara dS wycinka sfery jednostkowej odpowiadającego bliskim zera przyrostom $d\varphi$ i $d\vartheta$ współrzędnych sferycznych jest równa $\sin \vartheta \, d\vartheta \, d\varphi$. Stąd

$$\int_S x^i y^j z^k \, dS = \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi} (\cos \varphi \sin \vartheta)^i (\sin \varphi \sin \vartheta)^j \cos^k \vartheta \sin \vartheta \, d\vartheta \, d\varphi = A_{ij} B_{k,i+j+1},$$

gdzie $A_{ij} = \int_0^{2\pi} \cos^i \varphi \sin^j \varphi \, d\varphi$, $B_{k,l} = \int_0^{\pi} \cos^k \vartheta \sin^l \vartheta \, d\vartheta$.

268

Dość żmudne rachunki dają wzory

$$A_{00} = 2\pi, \quad A_{20} = A_{02} = \pi, \quad A_{40} = A_{04} = \frac{3}{4}\pi, \quad A_{22} = \frac{\pi}{4},$$

$$B_{01} = 2, \quad B_{03} = \frac{4}{3}, \quad B_{05} = \frac{16}{15}, \quad B_{21} = \frac{2}{3}, \quad B_{23} = \frac{4}{15}, \quad B_{41} = \frac{2}{5}.$$

Za ich pomocą można obliczyć m.in.

$$\|p_0\|^2 = 4\pi, \quad \|p_1\|^2 = \|p_2\|^2 = \|p_3\|^2 = \frac{4}{3}\pi, \quad \|p_4\|^2 = \|p_5\|^2 = \|p_6\|^2 = \frac{4}{15}\pi,$$

$$\|p_7\|^2 = \frac{16}{15}\pi, \quad \|p_8\|^2 = \frac{16}{5}\pi$$

i sprawdzić, że baza $\{p_0, \dots, p_8\}$ jest ortogonalna.

Zastosowana do znalezienia wielomianu p kwadratura bierze pod uwagę *wszystkie* tekstele tekstury irradiancji. Każdy tekstel jest kwadratem o boku $a = \frac{1}{\sqrt{2}}$ na ścianie kostki $[-1, 1]^3$. Rzut takiego kwadratu na sferę S jest wycinkiem o polu $\approx a^2 \cos^3 \alpha$, gdzie α jest kątem między odpowiadającym teksełowi wektorem w a wektorem normalnym ściany kostki.

Obliczenie wykona szader obliczeniowy, który realizuje trzy etapy: obliczenie składników kwadratur, sumowanie (parami) i końcowe dzielenie sum przez $\|p_i\|^2$ i zapisywanie wyników w miejscu docelowym.

Pokażę procedurę w GLSL-u realizującą pierwszy etap. Parametr ti określa miejsce wątku w (trójwymiarowej) grupie roboczej, parametr $size$ określa wymiary ściany kostki tekstury w teksełach.

```

switch ( ti.z ) {
case 0: w = vec3 ( 1.0, a, b ); break;
case 1: w = vec3 ( -1.0, a, b ); break;
case 2: w = vec3 ( a, 1.0, b ); break;
case 3: w = vec3 ( a, -1.0, b ); break;
case 4: w = vec3 ( a, b, 1.0 ); break;
case 5: w = vec3 ( a, b, -1.0 ); break;
}
w = normalize ( w );
ir = s*texture ( IrradianceTt, w ).rgb;
ind = 27*(ti.z*size + ti.y)*size + ti.x;
StoreVec ( ind, ir );
StoreVec ( ind+3, ir * w.x );
StoreVec ( ind+6, ir * w.y );
StoreVec ( ind+9, ir * w.z );
StoreVec ( ind+12, ir * (w.x*w.y) );
StoreVec ( ind+15, ir * (w.y*w.z) );
StoreVec ( ind+18, ir * (w.z*w.x) );
StoreVec ( ind+21, ir * ((w.x+w.y)*(w.x-w.y)) );
StoreVec ( ind+24, ir * ((w.z+w.x)*(w.z-w.x) + (w.z+w.y)*(w.z-w.y)) );
} /*Integrate*/

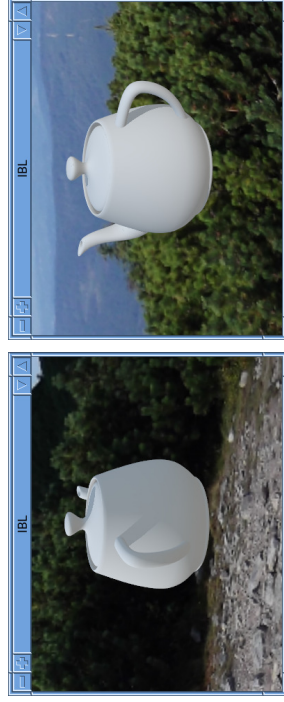
```

```

#version 450 core
layout (local_size_x=1) in;
layout (binding=0) uniform samplerCube IrradianceTt;
layout (location=0) uniform int n;
layout (binding=0, std430) buffer Irrad { float a[27]; } irrad;
layout (binding=1, std430) buffer Aux { float t[]; } aux;
void StoreVec ( uint ind, vec3 v )
{ aux.t[ind] = v.r; aux.t[ind+1] = v.g; aux.t[ind+2] = v.b; }
void Integrate ( uvec3 ti, uint size )
{
float a, b, s;
vec3 w, ir;
uint ind;
a = 2.0*((float)(ti.x)+0.5)/float(size))-1.0;
b = 2.0*((float)(ti.y)+0.5)/float(size))-1.0;
s = a*a + b*b + 1.0;
s = 4.0/float(size*size) * inversesqrt ( s ) / s;

```

Końcowy wynik:



273

Oświetlenie przez otoczenie powierzchni nielambertowskich

Radiancja światła odbitego od powierzchni nielambertowskiej, oświetlonej z wszystkich stron:

$$L_{rs}(\mathbf{p}, \mathbf{v}) = \int_{I \in S_{H^+}} \frac{DGF_\lambda}{4|\mathbf{l}, \mathbf{n}\rangle\langle \mathbf{v}, \mathbf{n}\rangle} L(\mathbf{p} + \mathbf{l}, -\mathbf{l})(\mathbf{l}, \mathbf{n}) dS. \quad (3)$$

W wyrażeniu podcałkowym wzór ten zastąpimy przybliżeniem, w którym czynniki D i $L(\mathbf{p} + \mathbf{l}, -\mathbf{l})$ zastępujemy przez ich wartość średnią:

$$L_{rs}(\mathbf{p}, \mathbf{v}) \approx \left(\frac{1}{2\pi} \int_{I \in S_{H^+}} DL(\mathbf{p} + \mathbf{l}, -\mathbf{l}) dS \right) \left(\int_{I \in S_{H^+}} \frac{GF_\lambda}{4|\mathbf{v}, \mathbf{n}\rangle} dS \right). \quad (4)$$

Czynnik $\frac{1}{2\pi}$ jest odwrotnością miary zbioru całkowania (tj. półsfery S_{H^+}). Pierwsza całka, L_i , opisuje przefiltrowaną radiancję światła dochodzącego z otoczenia. Druga całka, M_λ , odpowiada za odbijanie światła o określonej długości fali.

274

Biorąc parametryczne przedstawienie półsfery:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} (\lambda, \vartheta) = \begin{bmatrix} \cos \lambda \sin \vartheta \\ \sin \lambda \sin \vartheta \\ \cos \vartheta \end{bmatrix}, \quad \lambda \in [0, 2\pi), \vartheta \in [0, \pi/2],$$

możemy obliczyć całkę numerycznie:

$$\begin{aligned} \int_{I \in S_{H^+}} f(\mathbf{l}) dS &= \int_0^{\pi/2} \left(\int_0^{2\pi} f(\mathbf{l}(\lambda, \vartheta)) d\lambda \right) \sin \vartheta d\vartheta \\ &\approx \frac{\pi^2}{NM} \sum_{j=1}^M \sum_{i=1}^N f(\mathbf{l}(\lambda_i, \vartheta_j)) \sin \vartheta_j, \\ \lambda_i &= \pi \frac{2i-1}{N}, \vartheta_j = \pi \frac{2j-1}{4M}. \end{aligned}$$

Prostokąt $[0, 2\pi) \times [0, \pi/2]$ został podzielony na NM prostokątów, wszystkie współczynniki kwadratury mają tę samą wartość $\pi^2/(NM)$.

276

Dokonyamy preprocesingu, dokonując uproszczeń, aby oszczędzić miejsce na jego wyniki. Czynniki D zależy od wektorów \mathbf{n} i \mathbf{l} . Przyjmujemy, że zależy on tylko od kąta $\vartheta_{\mathbf{h}}$ między wektorami \mathbf{n} a \mathbf{h} . Oznacza to, że powierzchnia jest izotropowa.

Zapamiętamy (w tekście) odpowiednie całki tylko dla $\mathbf{v} = \mathbf{n}$. Wtedy wektor \mathbf{h} ma kierunek dwusiecznej kąta między wektorami \mathbf{l} a \mathbf{n} , a więc stabilizujemy funkcję $L_i(\mathbf{n})$, przyjmując kilka wartości parametru m , opisującego chropowatość powierzchni.

275

Mając liczby λ_i, ϑ_j , znajdujemy wektor $\vec{l} = (\cos \lambda_i \sin \vartheta_j, \sin \lambda_i \sin \vartheta_j, \cos \vartheta_j)$, potem poddajemy go odbiciu, które przeprowadza go na wektor \mathbf{e}_3 .

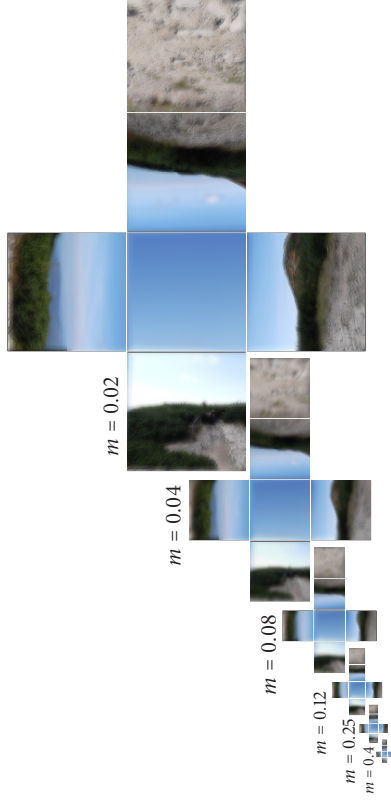
Na podstawie sinusa i kosinusa kąta ϑ_l między wektorami l a \mathbf{n} obliczamy

$$\cos^2 \vartheta_h = \frac{1 + \cos \vartheta_l}{2}, \quad \sin^2 \vartheta_h = \frac{\sin^2 \vartheta_l}{4 \cos^2 \vartheta_h}, \quad \operatorname{tg}^2 \vartheta_h = \frac{\sin^2 \vartheta_h}{\cos^2 \vartheta_h}.$$

Mając te liczby, można obliczyć czynnik D według przyjętego rozkładu (np. Beckmanna–Spizzichino).

Obliczenie możemy wykonać, rysując scenę na ścianach kostki.

277



278

Aby obliczyć całkę M_λ , czynnik Fresnela zastąpimy przybliżeniem Schlicka. Wtedy

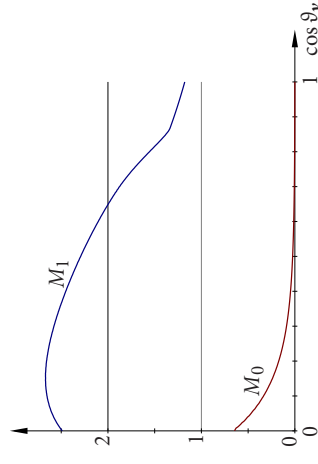
$$M_\lambda \approx \int_{I \in S_{n+}} \frac{G(F_{\lambda,0} + (1 - F_{\lambda,0})(1 - \langle l, \mathbf{h} \rangle)^5)}{4 \langle \mathbf{v}, \mathbf{n} \rangle} dS = M_0(\vartheta_v) + F_{\lambda,0} M_1(\vartheta_v), \quad (5)$$

$$\text{gdzie } M_0(\vartheta_v) = \int_{I \in S_{n+}} \frac{G(1 - \langle l, \mathbf{h} \rangle)^5}{4 \langle \mathbf{v}, \mathbf{n} \rangle} dS,$$

$$M_1(\vartheta_v) = \int_{I \in S_{n+}} \frac{G(1 - \langle l, \mathbf{h} \rangle)^5}{4 \langle \mathbf{v}, \mathbf{n} \rangle} dS.$$

Powyższe całki są funkcjami tylko jednego parametru, kąta ϑ_v między wektorami \mathbf{v} a \mathbf{n} . Stabilizowane całki, przedstawione jako funkcje kosinusa tego kąta, można zapamiętać w teksturze jednowymiarowej.

279



280

Funkcja L_i została obliczona przy założeniu $\mathbf{n} = \mathbf{v} = \mathbf{r}$, gdzie $-\mathbf{r}$ oznacza obraz wektora \mathbf{v} w odbiciu względem płaszczyzny stycznej do powierzchni. Wykonując końcowy obraz, jako argument poddawiamy wektor \mathbf{r} :

```
layout(binding=4, std430) buffer Irrad { float a[27]; } irradi;
layout(binding=10) uniform samplerCube SpecRadianceTxt;
layout(binding=11) uniform samplerID mOm1Txt;
vec3 SpecularEnvLighting ( vec3 normal, vec3 vv, float cthetav )
{
    vec3 Li, r, Fl;
    vec2 MOM1;
    int i, j, k;

    r = -reflect ( vv, normal );
    Li = texture ( SpecRadianceTxt, r ).rgb;
    MOM1 = texture ( mOm1Txt, cthetav ).xy;
    Fl = MOM1.xxx + MOM1.y*mm.F10;
    return Li*Fl;
} /*SpecularEnvLighting*/
```

281

```
vec3 PBRLighting ( void )
{
    vec3 lv, vv, Colour, rhoD, rhoS, lp;
    float dist, m2, cnv, cml, s;
    uint i, mask;

    vv = posDifference ( trb.eyepos, In.Position, dist );
    if ( dot ( vv, tnormal ) < 0.0 ) normal = -normal;
    cnv = dot ( vv, normal );
    Colour = mm.kD > 0.0 ?
        mm.kD*mm.cl*LightPoly ( normal ) : vec3 ( 0.0 );
    if ( mm.kS > 0.0 )
        Colour = mm.kS*SpecularEnvLighting ( normal, vv, cnv );
    m2 = mm.m*mm.m;
    for ( i = 0, mask = 0x00000001; i < light.nls; i++, mask <<= 1 )
        .... /* obliczanie oświetlenia przez źródła punktowe */
    return clamp ( Colour, 0.0, 1.0 );
} /*PBRLighting*/
```

W składniku opisującym światło dochodzące z otoczenia i odbite w sposób rozproszony model Orena-Nayara jest tu zastąpiony modelem Lamberta.

282

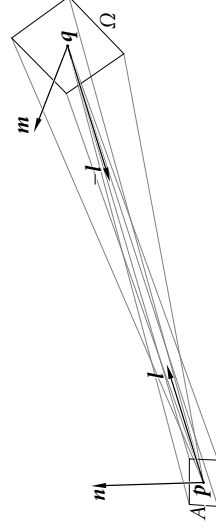
Funkcja L_i została obliczona przy założeniu $\mathbf{n} = \mathbf{v} = \mathbf{r}$, gdzie $-\mathbf{r}$ oznacza obraz wektora \mathbf{v} w odbiciu względem płaszczyzny stycznej do powierzchni. Wykonując końcowy obraz, jako argument poddawiamy wektor \mathbf{r} :

```
layout(binding=4, std430) buffer Irrad { float a[27]; } irradi;
layout(binding=10) uniform samplerCube SpecRadianceTxt;
layout(binding=11) uniform samplerID mOm1Txt;
vec3 SpecularEnvLighting ( vec3 normal, vec3 vv, float cthetav )
{
    vec3 Li, r, Fl;
    vec2 MOM1;
    int i, j, k;

    r = -reflect ( vv, normal );
    Li = texture ( SpecRadianceTxt, r ).rgb;
    MOM1 = texture ( mOm1Txt, cthetav ).xy;
    Fl = MOM1.xxx + MOM1.y*mm.F10;
    return Li*Fl;
} /*SpecularEnvLighting*/
```

281

Równanie bilansu energetycznego (ogólne)

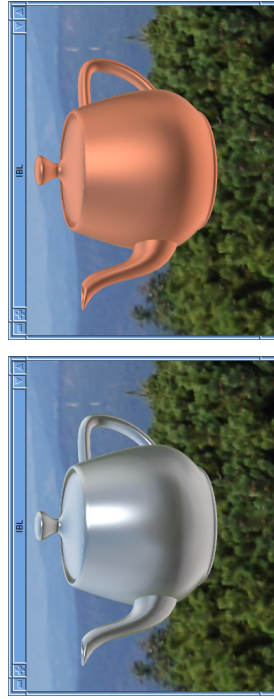


Strumień energetyczny światła dochodzącego z elementu powierzchni Ω do A , położonych w odległości r , dużo większej od ich średnic to

$$E = L(\mathbf{q}, -\mathbf{l})\Omega_s\psi_{qA},$$

gdzie $\mathbf{l} = \mathbf{q} - \mathbf{p}$, $L(\mathbf{q}, -\mathbf{l})$ oznacza radiancję światła wysłanego z punktu \mathbf{q} w kierunku $-\mathbf{l}$, Ω_s oznacza skróconą miarę elementu Ω widzianego z A , a ψ_{qA} oznacza kąt bryłowy elementu A widzianego z Ω .

284



283

Irradiancja elementu A światłem z Ω jest równa

$$I(\mathbf{p}, \mathbf{l}) = L(\mathbf{q}, -\mathbf{l})^2 \psi_{p\Omega} A |\cos \angle(\mathbf{n}, \mathbf{l})| / r^2 / A = L(\mathbf{q}, -\mathbf{l}) |\cos \angle(\mathbf{n}, \mathbf{l})| \psi_{p\Omega}.$$

Kąt bryłowy $\psi_{p\Omega}$ elementu Ω widzianego z A odpowiada elementowi dS sfery jednostkowej S , po której należy obliczyć całkę

$$L_r(\mathbf{p}, \mathbf{v}) = \int_{I \in S} \rho(\mathbf{p}, \mathbf{l}, \mathbf{v}) L(\mathbf{q}(\mathbf{p}, \mathbf{l}), -\mathbf{l}) |\cos \angle(\mathbf{n}, \mathbf{l})| dS,$$

która opisuje radiancję światła dochodzącego do punktu $\mathbf{p} \in A$ z wszystkich kierunków, które zostało odbite w kierunku wektora \mathbf{v} .

Symbol ρ oznacza dwukierunkową funkcję odbicia i załamania światła.

285

Całkowita radiancja światła opuszczającego punkt \mathbf{p} w kierunku \mathbf{v} jest sumą radiancji światła odbitego i radiancji światła wypromieniowanego przez ten punkt. Stąd wynika ogólne równanie bilansu energetycznego:

$$L(\mathbf{p}, \mathbf{v}) = L_e(\mathbf{p}, \mathbf{v}) + \int_{I \in S} \rho(\mathbf{p}, \mathbf{l}, \mathbf{v}) L(\mathbf{q}(\mathbf{p}, \mathbf{l}), -\mathbf{l}) |\cos \angle(\mathbf{n}, \mathbf{l})| dS. \quad (6)$$

Jest tu przyjęte założenie, że ośrodki, przez które przenika światło, są całkowicie przezroczyste. Wartością funkcji $\mathbf{q}(\mathbf{p}, \mathbf{l})$ jest punkt powierzchni widoczny z \mathbf{p} w kierunku \mathbf{l} .

286

Równanie bilansu energetycznego (przypadek lambertowski)

W modelu Lamberta zakłada się, że powierzchnie są nieprzezroczyste, idealnie gładkie i matowe. Dla ustalonej długości fali λ dwukierunkowa funkcja (załamania i) odbicia światła w każdym punkcie \mathbf{p} przyjmuje dwie wartości:

$$\rho(\mathbf{p}, \mathbf{l}, \mathbf{v}) = \begin{cases} c_\lambda(\mathbf{p}) \in [0, 1/\pi] & \text{jeśli } \langle \mathbf{l}, \mathbf{n} \rangle \text{ i } \langle \mathbf{v}, \mathbf{n} \rangle \text{ mają ten sam znak,} \\ 0 & \text{w przeciwnym razie.} \end{cases}$$

Przy założeniu, że widoczna jest tylko jedna strona każdej powierzchni, równanie (6) można zatem uprościć do

$$L(\mathbf{p}) = L_e(\mathbf{p}) + \rho(\mathbf{p}) \int_{I \in S_{H^+}} L(\mathbf{q}(\mathbf{p}, \mathbf{l})) \cos \angle(\mathbf{n}, \mathbf{l}) dS, \quad (7)$$

z całką po półsferyze $S_{H^+} = \{I \in S: \langle \mathbf{l}, \mathbf{n} \rangle \geq 0\}$.

287

Zamiast po półsferyze, można całkować po wszystkich powierzchniach sceny. Niech $\mathbf{q} = \mathbf{q}(\mathbf{p}, \mathbf{l})$ oznacza punkt elementu powierzchni $d\Omega$.

Kąt bryłowy tego elementu to

$$d\Omega = \frac{|\cos \angle(\mathbf{m}, \mathbf{p} - \mathbf{q})|}{\|\mathbf{p} - \mathbf{q}\|^2} d\Omega,$$

ale ponieważ nie cały element $d\Omega$ musi być widoczny z punktu \mathbf{p} , trzeba wprowadzić czynnik $v(\mathbf{p}, \mathbf{q})$ równy 1, jeśli punkty \mathbf{p} i \mathbf{q} „widzą się” nawzajem i 0 w przeciwnym razie.

Równanie bilansu energetycznego dla przypadku lambertowskiego ma postać

$$L(\mathbf{p}) = L_e(\mathbf{p}) + \rho(\mathbf{p}) \int_{\mathbf{q} \in \Omega} v(\mathbf{p}, \mathbf{q}) \frac{\cos \angle(\mathbf{n}, \mathbf{q} - \mathbf{p}) \cos \angle(\mathbf{m}, \mathbf{p} - \mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|^2} L(\mathbf{q}) d\Omega. \quad (8)$$

288

W postaci operatorowej:

$$L = L_e + \mathcal{K}L \quad \text{albo} \quad (1 - \mathcal{K})L = L_e.$$

Przy założeniu, że powierzchnie sceny można podzielić (z góry) skończenie wieloma gładkimi krzywymi na kawałki, w których funkcje ρ i L_e są ciągłe i rozszerzają się w sposób ciągły na brzegi tych kawałków, a ponadto funkcja ρ ma maksymalną wartość mniejszą od $1/\pi$, jest $\|\mathcal{K}\|_\infty < 1$ i szereg Neumanna

$$L_e + \mathcal{K}L_e + \mathcal{K}^2L_e + \mathcal{K}^3L_e + \dots$$

jest zbieżny do kawałkami ciągłej funkcji L .

289

Dyskretyzacja lambertowskiego równania bilansu energetycznego

Powierzchnie sceny podzielę na n elementów o małych średnicach i będę poszukiwać funkcji L_h stałej w każdym elemencie.

Jeśli ϕ_i oznacza funkcję równą 1 w i -tym elemencie i 0 we wszystkich pozostałych, to liczby L_i , takie że

$$L_h = \sum_{i=1}^n L_i \phi_i$$

są wartościami funkcji L_h w elementach.

Funkcję L_e zastąpię jej przybliżeniem

$$L_{eh} = \sum_{i=1}^n L_{ei} \phi_i.$$

290

Rozpatruję dwie metody dyskretyzacji.

Metoda kolokacji: w każdym elemencie wybieram jeden punkt, \mathbf{p}_i (np. środek ciężkości). Zakładam, że równanie (8) jest spełnione w tych punktach. Wtedy powstaje układ równań

$$L_i = L_{ei} + \rho(\mathbf{p}_i) \sum_{j=1}^n \left(\int_{q \in A_j} v(\mathbf{p}_i, \mathbf{q}) \frac{\cos \angle(\mathbf{n}_i, \mathbf{q} - \mathbf{p}_i) \cos \angle(\mathbf{n}_j, \mathbf{p}_i - \mathbf{q})}{\|\mathbf{p}_i - \mathbf{q}\|^2} d\Omega \right) L_j, \quad i = 1, \dots, n.$$

Wartości całki oznaczę symbolami G_{ij} . Mam więc układ

$$L_i = L_{ei} + \rho(\mathbf{p}_i) \sum_{j=1}^n G_{ij} L_j, \quad i = 1, \dots, n.$$

291

Metoda Galerkin: residuum równania (8) ma mieć w każdym elemencie wartość średnią 0. Po scałkowaniu wynika stąd równanie

$$L_i = \frac{1}{A_i} \int_{\mathbf{p} \in A_i} L_e(\mathbf{p}) + \rho(\mathbf{p}) \sum_{j=1}^n \left(\int_{q \in A_j} v(\mathbf{p}, \mathbf{q}) \frac{\cos \angle(\mathbf{n}_i, \mathbf{q} - \mathbf{p}) \cos \angle(\mathbf{n}_j, \mathbf{p} - \mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|^2} d\Omega \right) L_j d\Omega.$$

Zastąpię funkcje ρ oraz L_e przez ich wartości średnie w elemencie A_i :

$$\rho_i = \frac{1}{A_i} \int_{\mathbf{p} \in A_i} \rho(\mathbf{p}) d\Omega, \quad L_{ei} = \frac{1}{A_i} \int_{\mathbf{p} \in A_i} L_e(\mathbf{p}) d\Omega,$$

Otrzymam (jeszcze trochę inne) równanie

$$L_i = L_{ei} + \rho_i \sum_{j=1}^n \left(\frac{1}{A_i} \int_{\mathbf{p} \in A_i} \int_{q \in A_j} v(\mathbf{p}, \mathbf{q}) \frac{\cos \angle(\mathbf{n}_i, \mathbf{q} - \mathbf{p}) \cos \angle(\mathbf{n}_j, \mathbf{p} - \mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|^2} d\Omega d\Omega \right) L_j.$$

292

Powstaje stąd układ równan liniowych

$$L_i = L_e + \rho_i \sum_{j=1}^n F_{ij} L_j, \quad i = 1, \dots, n.$$

Liczby F_{ij} są nazywane **współczynnikami kształtu** (ang. *form factors*).

Dla każdego i, j jest $A_i F_{ij} = A_j F_{ji}$, a ponadto $F_{ii} = 0$.

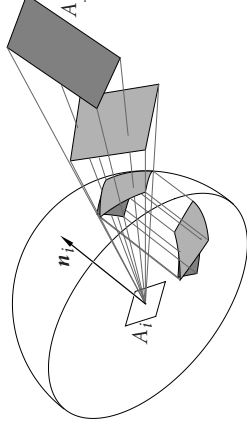
Na ogół $A_i G_{ij} \neq A_j G_{ji}$, a ponadto może być $F_{ij} > 0 = G_{ij}$, ale liczby G_{ij} można uznać za przybliżenia współczynników kształtu: zewnętrzna całka w definicji F_{ij} jest zastąpiona przez kwadraturnę opartą na jednym węźle — punkcie kolokacji.

Przechodząc (ponownie) do całkowania po kierunkach, otrzymamy wzór

$$\begin{aligned} G_{ij} &= \int_{q \in A_j} v(\mathbf{p}_i, \mathbf{q}) \frac{\cos \angle(\mathbf{n}_i, \mathbf{q} - \mathbf{p}_i) \cos \angle(\mathbf{n}_j, \mathbf{p}_i - \mathbf{q})}{\|\mathbf{p}_i - \mathbf{q}\|^2} d\Omega \\ &= \int_{T \in S_j} \cos \angle(\mathbf{n}_i, \mathbf{T}) dS. \end{aligned}$$

293

Symbol S_j oznacza wycinek półsfery S_{n+} składających się z tych wektorów \mathbf{l} , w których z punktu \mathbf{p}_i widoczny jest element A_j . Ten sposób obliczania współczynników kształtu jest nazywany **analogią Nusselta**.



Dla każdego i pola podwójnych rzutów elementów A_j wypełniają koło o promieniu 1, zatem dla każdego i suma liczb G_{ij} jest nie większa niż π .

294

Zdyskretyzowany operator całkowy \mathcal{K} reprezentują przez iloczyn DF macierzy diagonalnej D , której współczynnikami są liczby ρ_i , oraz macierzy współczynników kształtu F . Dla ułatwienia, macierz D pomnożę przez π (jej współczynniki diagonalne są liczbami z przedziału $[0, 1)$), a macierz F pomnożę przez $1/\pi$ (w każdym wierszu suma jej współczynników nie przekracza 1).

Wtedy proces iteracyjny

$$L_k = L_e + DF L_{k-1}, \quad k = 1, 2, 3, \dots, \quad (9)$$

wytwarza szereg Neumanna zbieżny do rozwiązania układu równań liniowych, które chcę znaleźć. Trzeba to zrobić dla kilku (np. trzech) długości fali świetlnej.

Scena może być oświetlona także (albo tylko) przez punktowe źródła światła. Można je zamienić na światło emitowane przez powierzchnie — wystarczy pomnożyć irradiancję przez wartość funkcji ρ i dodać to do wartości funkcji L_{eh} .

295

Wektor L_K otrzymany po wykonaniu K iteracji służy do utworzenia **tekstury irradiancji**. Opisuje ona, dla każdego punktu powierzchni sceny, (przybliżoną) irradiancję tego punktu światłem dochodzącym z otoczenia. Po pomnożeniu przez wartość funkcji ρ w danym punkcie i dodaniu irradiancji światła emitowanego przez dany punkt, otrzymamy radiancję, której można użyć do wykonania końcowego obrazu.

296

Elementy implementacji

Scena (tj. zbiór powierzchni odbijających światło) składa się z trójkątów o różnych rozmiarach. Duże trójkąty trzeba podzielić na odpowiednio małe elementy, a bardzo małe trójkąty lepiej jest połączyć w większe zespoły, które dalej będą podzielone na elementy dyskretyzacji.

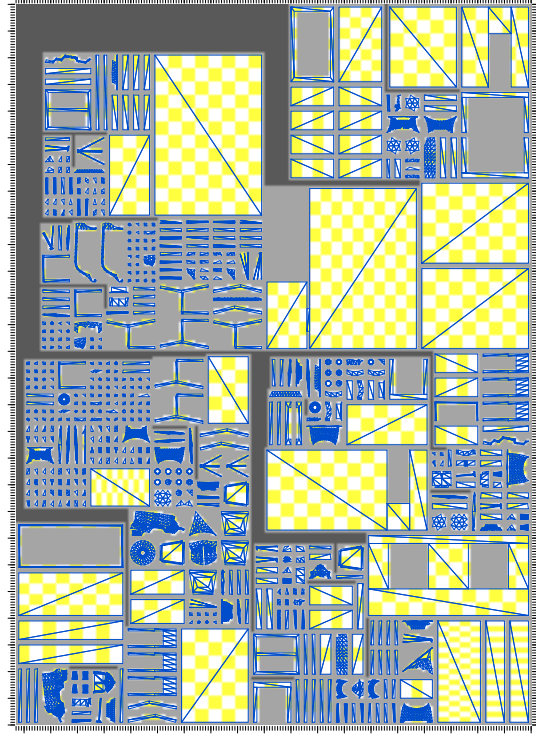
Wspomniane zespoły nazwałem **płatami**; płat składa się z trójkątów połączonych wspólnymi krawędziami lub wierzchołkami i takich, że można dokonać rzutu prostopadłego wszystkich trójkątów na płaszczyznę, otrzymując trójkąty o rozłącznych wnętrzach (trójkąty w przestrzeni nie muszą być ściśle współpłaszczyznowe).

297

Obrazy płatów w rzutach na płaszczyznę będą odpowiednio przeskalowane, obrócone i rozmieszczone na płaszczyźnie tak, aby możliwe ciasno upakowane zmieściły się w pewnym prostokącie o wymiarach $w \times h$. Prostokąt ten będzie dziedziną tekstury irradiancji; kwadraty 1×1 (teksele) wyznaczają elementy dyskretyzacji, po dokonaniu odwzorowania odwrotnego.

Wprowadzilem też **makroelementy** — nieco większe zespoły elementów w obrębie plata. Umożliwiają one zmniejszenie ilości potrzebnej pamięci i zmniejszają też błędy wynikające z niedokładnego obliczania współczynników kształtu G_{ij} .

298



299

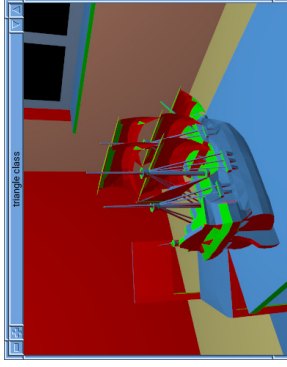
Preprocessing na CPU

Aplikacja wprowadza do struktury danych reprezentacje poszczególnych obiektów w postaci tablic wierzchołków i trójkątów. Trójkąty powinny tworzyć powierzchnie zamknięte, tj. brzozy brył wielościennych, a ich orientacja ma prowadzić do obliczenia wektorów normalnych skierowanych na zewnątrz tych brył.

Dla każdego obiektu znajdowane są platy. W tym celu trójkąty są klasyfikowane — każda klasa odpowiada pewnemu wektorowi jednostkowemu. Do danej klasy są zaliczane trójkąty, których wektor normalny tworzy z danym wektorem najmniejszy kąt. Przyjęte 6 klas odpowiada wektorom $e_1, -e_1, e_2, -e_2, e_3, -e_3$.

Można też wprowadzać zbiory trójkątów z zaznaczeniem, że mają z nich powstać platy z pominięciem klasyfikacji.

300

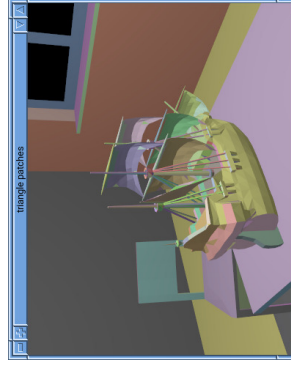


301

Podział klasy na płaty odbywa się przy użyciu algorytmu łączenia drzew według rangi w lesie zbiorów rozłącznych. **Graf sąsiedztwa trójkątów** ma wierzchołki — trójkąty — i krawędzie wyznaczone przez wspólne wierzchołki. **Graf pomocniczy** jest lasem, który ma początkowo tylko wierzchołki. Jest on przetwarzany w celu znalezienia składowych spójnych grafu sąsiedztwa trójkątów. Po wykryciu wspólnego wierzchołka trójkątów i sprawdzeniu, że należą one do różnych drzew w lesie, jedno z tych drzew jest podczepiane pod korzeń drugiego.

W wyniku, dla każdego trójkąta jest znajdowany identyfikator jego składowej spójnej, po czym można tablicę trójkątów posortować według tych identyfikatorów i w ten sposób znaleźć płaty.

302



303

Dla płata jest znajdowany wektor normalny płaszczyzny, na którą będzie dokonane rzutowanie oraz skalowanie, które ustali wielkość elementów. Wierzchołki trójkątów są rzutowane na tę płaszczyznę.

Następnie (za pomocą algorytmu Grahama) jest znajdowana otoczka wypukła zbioru rzutów wierzchołków trójkątów płata oraz taki jej obrót, aby otoczka została wpisana w najmniejszy prostokąt o bokach poziomym i pionowym. Jeśli szerokość prostokąta jest mniejsza niż wysokość, to te wymiary są przestawiane.

Dalsze etapy preprocesingu są wykonywane po przepisaniu danych opisujących wszystkie obiekty (ich trójkąty i płaty) do tablic wspólnych.

304

Aby rozmieścić (i upakować) płyty w prostokącie, który będzie dziedziną tekstury, tworzony jest las drzew binarnych, początkowo bez krawędzi, którego wierzchołkami są płyty. Wierzchołki są wstawiane do kolejki priorytetowej (kopca), przy czym większy priorytet ma wierzchołek wpisany w prostokąt o mniejszej szerokości, a jeśli szerokości są równe, to w ten o mniejszej wysokości.

Węzły, z kolejki są wyjmowane dwa wierzchołki (korzenie drzew w lesie), po czym jest tworzony nowy wierzchołek, z prostokątem obejmującym prostokąty wyjętych wierzchołków, które zostają poddrzewami tego nowego wierzchołka. Jeśli szerokość jest mniejsza niż wysokość, to te wymiary są przestawiane. Nowy wierzchołek jest wstawiany do kolejki. Proces kończy się, gdy powstanie jedno drzewo binarne.

Nadawanie wierzchołkom trójkątów położen w dziedzinie tekstury irradiancji odbywa się podczas przeszukiwania tego drzewa w głąb.

305

Preprocessing na GPU

Za pomocą napisanych w GLSL-u programów działających na procesorze graficznym, wszystkie trójkąty są rysowane w pozaekranowym buforze ramki, którego wymiary w, h odpowiadają wymiarom tekstury irradiancji. Najpierw są rysowane krawędzie trójkątów, a potem wypełniane wnętrza. W ten sposób dla każdego tekstela jest określane, czy ma on być „używany”, tj. ma mu odpowiadać element dyskretyzacji, czy nie.

Informacje wytworzone w preprocessingu są przechowywane w tablicach, przy czym są dwa podstawowe sposoby dostępu do nich: **tablica indeksowana współrzędnymi tekstela** przechowuje dane dla wszystkich teksteli (w tym „nieużywanych”). Indeks jest obliczany ze wzoru $i = wy + x$, gdzie x, y to współrzędne całkowite tekstela.

Tablica indeksowana numerem elementu przechowuje informacje związane z tekstelami „używanymi”, jej indeks od 0 do $n - 1$ jest numerem zmiennej w układzie równań.

306

Aby określić makroelementy, rysowane są prostokąty opisane na płytach. Każdy z nich jest pokrywany teksturą szachownicy. Z numeru płyta i numerów pól we wzorze szachownicy są tworzone liczby całkowite — tymczasowe identyfikatory makroelementów. Teksele nieużywane dostają numer makroelementu $2^{32} - 1$. Następnie tablica z tekstelami jest sortowana w kolejności rosnących identyfikatorów. W wyniku tekstele należące do każdego makroelementu znajdują się w tablicy obok siebie, a tekstele nieużywane na końcu.

Sortowanie jest wykonywane za pomocą sieci sortującej, w $O(\log^2 n)$ krokach. Sortowanie powoduje, że tablica teksteli, początkowo indeksowana współrzędnymi tekstela, będzie indeksowana numerem elementu.

Dla posortowanego ciągu elementów tworzony jest ciąg zer i jedynek: jedynka odpowiada pierwszemu elementowi należącemu do kolejnego makroelementu. Ciąg sum prefiksowych tego ciągu wytwarza ostateczne numery makroelementów.

308

Dla każdego tekstela, który należy do obrazu pewnego trójkąta (czyli został początkowo oznaczony jako „używany”) jest wyznaczane jego przecięcie z rzutem trójkąta — algorytmem obcinania Sutherlanda–Hodgmana. Jeśli pole przecięcia jest zbyt małe, to tekstel zostaje oznaczony jako nieużywany.

W przeciwnym razie szader oblicza środek ciężkości wierzchołków przecięcia. Jego przeciwobraz na trójkącie w przestrzeni będzie przyjęty jako punkt kolokacji.

Używane tekstele dostają dodatkowy atrybut równy 1, a nieużywane — 0.

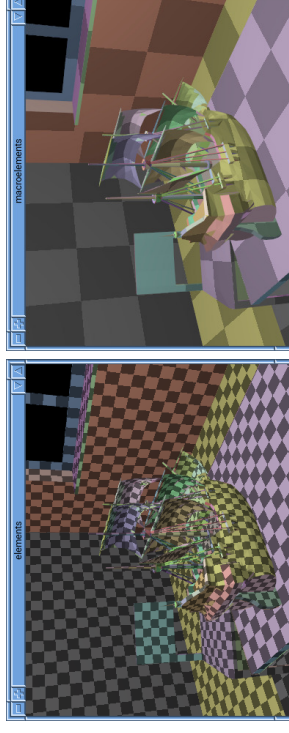
Suma tych atrybutów (obliczona przez równoległy algorytm sumowania parami) jest liczbą n elementów dyskretyzacji.

307

Dla każdego elementu dyskretyzacji jest przechowywana informacja o numerze trójkąta, numerze makroelementu i współrzędne x, y tekstela w dziedzinie tekstury.

Dla każdego używanego tekstela jest przechowywana informacja o numerze $i \in \{0, \dots, n - 1\}$ elementu.

309



310

Macierz F współczynników kształtu będzie zastąpiona przez przybliżający ją iloczyn GA dwóch macierzy, $G \in \mathbb{R}^{n \times m}$ i $A \in \mathbb{R}^{m \times n}$, gdzie m oznacza liczbę makroelementów.

Dla wektora $\mathbf{x} \in \mathbb{R}^n$ j -ta współrzędna wektora $A\mathbf{x}$ jest średnią ważoną

współrzędnych wektora \mathbf{x} odpowiadających elementom j -tego makroelementu.

Wagi są proporcjonalne do pól przecięć teksteli z obrazami odpowiednich trójkątów. W każdej kolumnie ma ona zatem jeden niezerowy współczynnik.

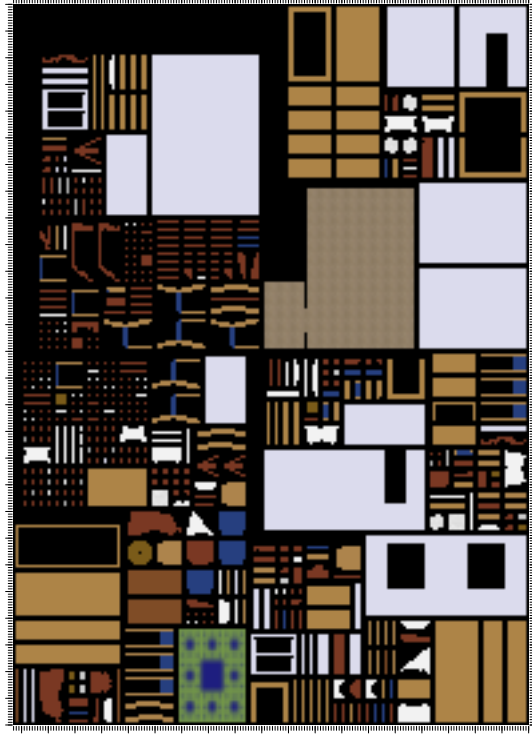
Macierz G składa się ze współczynników kształtu obliczonych w sposób opisany dalej, z tym, że współczynnik G_{ij} odpowiada elementowi A_i i makroelementowi M_j . To też jest macierz rzadka, ale rozmieszczenie jej niezerowych współczynników jest całkowicie nieregularne.

311

Jest jeszcze macierz diagonalna D , której współczynniki są (pomnożonymi przez π) uśrednionymi wartościami funkcji ρ w poszczególnych elementach. Oblicza się je, rysując trójkąt w dziedzinie tekstury i nadając tekstelom kolory otrzymane z opisu materiału poszczególnych trójkątów, który może być reprezentowany za pomocą tekstury. Za uśrednianie odpowiada wbudowany w proces rysowania przez GPU podsystem filtrowania tekstur.

W tablicy o długości n są przechowywane trójki liczb rzeczywistych — poszczególne liczby opisują odbijanie światła czerwonego, zielonego i niebieskiego. Są to więc trzy macierze D , dla trzech kolorów podstawowych.

312



313

Obliczanie współczynników kształtu

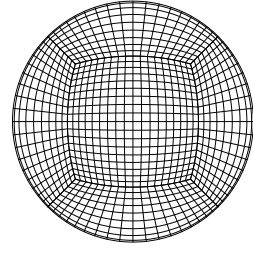
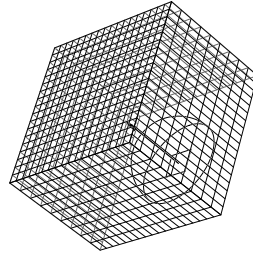
Pole obrazu widocznej części makroelementu M_j widzianego z punktu kolokacji P_i na elemencie A_i , rzutowanego na półsferę jednostkową a następnie na koło mające z półsferą wspólny brzeg, jest obliczane przez numeryczne całkowanie funkcji charakterystycznej tego obrazu. Do tego przydaje się GPU, realizująca program rysowania sceny na pięciu ścianach kostki „obudowanej” nad płaszczyzną elementu A_i .

Powstają obrazy rastrowe; każdy piksel jest „zamalowany” numerem widocznego w nim makroelementu.

Wagi pikseli, którym został przypisany ten sam numer j , zostaną zsumowane.

Waga jest proporcjonalna do pola czworokąta krzywoliniowego, który jest obrazem w podwójnym rzucie pikseli na ścianie kostki.

314



Górna ściana kostki jest kwadratem 2×2 . Wysokość kostki $C \approx 1.543$ jest dobrana tak, aby zminimalizować sumę kwadratów odchyłek wag pikseli od wagi średniej.

315

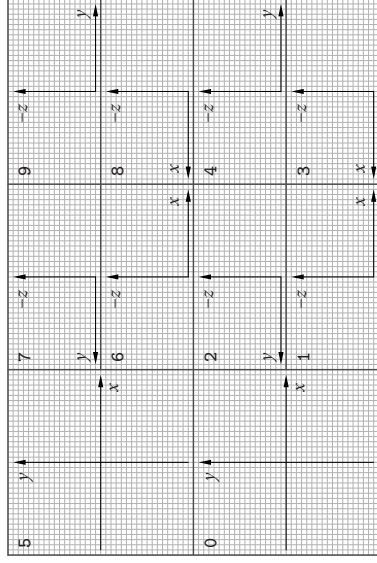
Dla każdego elementu jest wykonywane 10 obrazów w rzutach perspektywicznych; środkiem rzutowania jest punkt kolokacji na elemencie. Dla każdej ściany kostki są wykonywane dwa obrazy, odpowiadające różnym zakresom odległości. Ma to na celu poprawienie dokładności algorytmu rozstrzygnięcia widoczności, w którym głębokości punktów są reprezentowane przez liczby typu float.

Te 10 obrazów jest wykonywane jednocześnie.

OpenGL umożliwia wykonanie obrazów jednocześnie nawet w 16 klatkach.

Klatki mają wymiary 36×36 i 36×18 pikseli. Taka rozdzielczość obrazów mieści się w limicie wynikającym z ilości dostępnej pamięci podręcznej GPU.

316



317

Obrazy dla każdego elementu składają się z 3888 pikseli; w każdym z nich jest zapisany numer widocznego makroelementu. Dwie tablice o długości 3888 w pamięci podręcznej GPU (zadeklarowane z kwalifikatorem `shared`) przechowują numery makroelementów (liczby typu `uint`) i wagi pikseli (typu `float`) i razem zajmują 31104 bajty (limit to 32 kB).

Z tych danych konstruuje się jeden wiersz macierzy G . Najważniejszą pracą wykonuje szader obliczeniowy, którego lokalna grupa robocza składa się z $972 = 3888/4$ wątków (standard zapewnia limit 1024 wątków). Każdy wątek musi zatem przetworzyć kilka pikseli. Obliczenie współczynników kształtu następuje w jednym wywołaniu grupy roboczej.

Pierwszym krokiem jest przepisanie z bufora do pamięci podręcznej numerów makroelementów i wag pikseli.

W drugim kroku numery makroelementów razem z wagami pikseli są sortowane (algorytmem sieci sortującej), a potem przepisywane z powrotem do buforów.

318

Obliczenie macierzy G jest podzielone na etapy, w których oblicza się bloki składające się z 1024 wierszy. Po znalezieniu wszystkich bloków składa się je w całość.

Szader obliczeniowy, który oblicza współczynniki kształtu, pracuje w lokalnych grupach jednowątkowych i jest wywoływany kilkakrotnie, aby zrealizować kolejne etapy obliczenia.

Mając liczby niezerowych współczynników w każdym wierszu, można obliczyć sumy prefiksowe i wpisać je do tablicy r .

W ostatnim etapie każdy wątek szadera ma za zadanie obliczenie jednego współczynnika kształtu, przez zsumowanie wag. Użyty tu jest sekwencyjny algorytm sumowania. Obliczenie kończy wpisanie wyników do tablic r i c .

320

Do pamięci podręcznej są wpisywane zera i jedyńki; jedynka trafia na miejsce odpowiadające pierwszemu wystąpieniu numeru makroelementu w posortowanym ciągu.

Następnie szader oblicza sumy prefiksowe tego ciągu zer i jedynek i przepisuje je do przeznaczzonego na to bufora. W ten sposób określone są podciągi wag do zsumowania i w szczególności liczba niezerowych współczynników w wierszu macierzy G .

Sumowanie wag wykona następny szader obliczeniowy.

319

Obliczanie tekstury irradiancji

Przed przystąpieniem do iterowania wzoru (9) trzeba obliczyć wektor $L_0 = L_e$. W tym celu trójkąt jeszcze raz rysuje się w dziedzinie tekstury irradiancji, a raczej w prostokącie $s = 4$ razy większym, obliczając ich oświetlenie przez punktowe źródła światła. Dla każdego piksela jest obliczana suma radiancji światła odbitego i światła emitowanego przez trójkąt, dla składowych r , g , b .

Radiancja światła emitowanego przez każdy element dyskretyzacji jest obliczana jako średnia radiancja s^2 pikseli otrzymanego wyżej obrazu.

321



322

Po wykonaniu K iteracji, przy użyciu szaderów obliczeniowych, które realizują mnożenie macierzy rzadkich A , G i D przez wektor L_{k-1} i dodawanie wektorów, składowe wektora L_K są używane do otrzymania wartości teksteli tekstury irradiancji — przez obliczenie iloczynu $GA L_K$ (już bez mnożenia przez macierz D).

Teksele nieużywane, które sąsiadują z tekstelami używanymi, otrzymują wartości średnie tych teksteli; bez tego, podczas wykonywania końcowych obrazów sceny, ewaluator, który dokonuje interpolacji i filtrowania tekstury siegalby po nieokreślone wartości tych teksteli, dając błędne obrazy.

323



324

Kolory pikseli na końcowych obrazach są obliczane zgodnie z modelem Lamberta, na podstawie bezpośredniego oświetlenia ze źródeł punktowych, emisji własnej powierzchni oraz przefiltrowanej irradancji z tekstury irradiancji. Wartość funkcji ρ w danym punkcie jest brana z opisu materiału powierzchni, której to jest punkt.

325

Przykładowe wyniki

dyskretyzacja	n	m	$w \times h$	N
zgrubna	12758	1246	200×158	2413743
średnia	26382	1858	270×193	7238061
drobna	97893	5215	482×340	73905549
b. drobna	171074	14694	653×441	199344607

W pierwszych trzech przypadkach macierz G ma ok 15% niezerowych współczynników, a w czwartym tylko ok. 8%. Jest to spowodowane małą rozdzielczością obrazów na ścianach kostki.

326

	t_{prep}	t_G	t_{draw}	t_{solve}	t_1	t_2	t_3
RTX 3060	zgrubna	0.901	0.893	0.464	0.1018	0.0022	0.0926
	średnia	1.848	1.836	0.959	0.1793	0.0032	0.1638
	drobna	7.240	7.200	3.581	1.3560	0.0051	1.2317
	b. drobna	13.307	13.228	6.288	3.6621	0.0079	3.3244
GTX 940M	zgrubna	10.170	10.134	7.916	1.1170	0.0052	1.0099
	średnia	21.240	21.176	16.358	3.3478	0.0078	3.0353

t_{prep} — czas preprocessingu, w tym

t_G — czas obliczania współczynników kształtu, w tym

t_{draw} — całkowity czas rysowania sceny na ścianach kostki.

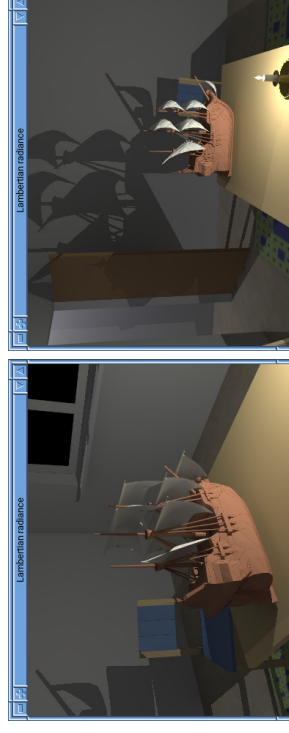
t_{solve} — czas rozwiązywania układu równań. Jego składniki to:

t_1 — czas obliczania wektora L_e ,

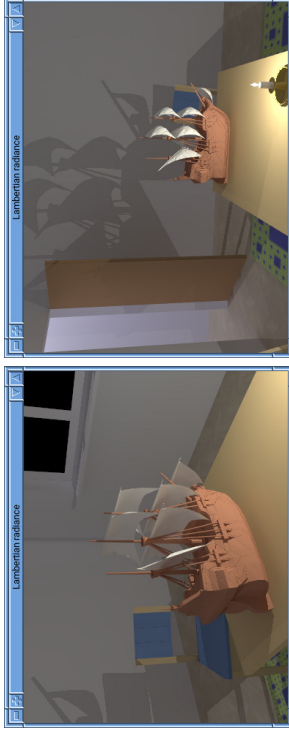
t_2 — czas wykonywania I0 iteracji,

t_3 — czas nadawania teksełom końcowych wartości.

327



328



329

Opóźnione cieniowanie

Opóźnione cieniowanie (wng. *deferred shading*) jest techniką, w której obraz jest wykonywany w dwóch etapach. W pierwszym następuje obcinanie, rasteryzacja i rozstrzyganie widoczności, po którym dla każdego piksela są zapamiętywane informacje o widocznym w tym pikselu obiekcie. Obliczanie koloru piksela następuje w etapie drugim na podstawie tych informacji.

Ma to dwie zalety: obliczenie koloru, które może być czasochłonne, jest wykonywane dla każdego piksela tylko raz. Po drugie, między tymi etapami można użyć dowolnych technik przetwarzania obrazu dla uzyskania efektów specjalnych. W szczególności jest wtedy dostępna informacja na temat tego, co widać w pikselach sąsiednich.

Wadą tej techniki jest niemożność wykonania antyaliasingu przez wielopróbkowanie. Można wykonać obraz o większej rozdzielczości, aby przeprowadzić nadpróbkowanie.

330

Inną wadą jest spora (nawet kilkanaście razy większa) ilość pamięci potrzebna do przechowania informacji uzyskanych w pierwszym etapie. Dla każdego piksela trzeba przechować przykładowo:

- wektor współrzędnych położenia punktu w układzie świata,
- wektory normalne powierzchni i płaszczyzny przybliżającego ją trójkąta,
- wektor współrzędnych tekstury,
- identyfikator obiektu, prymitywu (np. trójkąta) lub materiału.

Zestaw tablic, w których przechowuje się te informacje, jest nazywany **G-buforem**.

331

Aby utworzyć G-bufor w aplikacji OpenGL-a, trzeba do pozaekranowego bufora ramki dołączyć tekstury — załączniki `GL_COLOR_ATTACHMENT0, ..., GL_COLOR_ATTACHMENTn` (oraz bufor głębokości, `GL_DEPTH_ATTACHMENT`).

Procedura `glDrawBuffers` służy do określenia, które załączniki mają być używane, a w treści szadera fragmentów pierwszego etapu rysowania zmienne wyjściowe poprzedza się kwalifikatorem

`layout(location=k) out zmienna_wyjsciowa;`

W drugim etapie trzeba narysować prostokąt o wymiarach klatki (czyli kwadrat $[-1, 1] \times [-1, 1] \times \{0\}$). Shader fragmentów otrzymuje współrzędne piksela, którego kolor ma obliczyć. Obrazy tekstur będących załącznikami G-bufora muszą być mu udostępnione za pomocą procedury `glBindImageTexture`.

332

W zasadzie można na podstawie współrzędnych (ξ, η) piksela i głębokości ζ widocznego w tym pikselu punktu P odtworzyć położenie tego punktu (wektor współrzędnych w układzie świata). Zobaczymy, jak to zrobić, a potem zastanówmy się, dlaczego nie warto.

Jeśli klatka ma szerokość w , wysokość h i dolny lewy wierzchołek w punkcie (ξ_l, η_b) , to możemy obliczyć wektor współrzędnych w układzie ekstoki standardowej

$$(x, y, z) = (2(\xi + 0.5 - \xi_l)/w - 1, 2(\eta + 0.5 - \eta_b)/h - 1, 2\zeta - 1).$$

Wektor $Q = (x, y, z, 1)$ trzeba pomnożyć przez odwrotność iloczynu macierzy V i P (opisującego przejście od układu świata do kostki standardowej), otrzymując wektor $P = (PV)^{-1}Q$ współrzędnych jednorodnych punktu P .

Błędy zaokrągleń popelnionych w tym obliczeniu powodują niedokładność wyniku. Dla obliczeń oświetlenia (tj. wektorów kierunków do źródeł światła o obserwatora) to jest mało istotne, ale dla algorytmu cieni to może mieć bardzo duży (i niekorzystny) wpływ na wynik.

333

Obrazowanie poświaty

Wokół świecących w ciemnym otoczeniu lamp i płomieni (np. świecy) powstaje poświata. Przed wykonaniem końcowego obrazu można obrazy świecących obiektów „rozmyć”, aby otrzymać ten efekt.

Funkcja rozkładu normalnego Gaussa o odchyleniu standardowym σ jest dana wzorem

$$\mathcal{N}_\sigma(x) \stackrel{\text{def}}{=} \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}.$$

Funkcja ta jest parzysta, tj. $\mathcal{N}_\sigma(-x) = \mathcal{N}_\sigma(x)$ dla każdego x , przyjmuje maksymalną wartość dla $x = 0$ i ze wzrostem $|x|$ maleje do zera, przy czym w wielu zastosowaniach praktycznych (także w tu opisanym) dla $|x| > 3\sigma$ jej wartości są zanedbywalnie małe. Całka z funkcji \mathcal{N}_σ po całym zbiorze liczb rzeczywistych jest równa 1.

334

Filtr potrzebny do przetwarzania obrazów jest funkcją dwóch zmiennych; otrzymamy go ze wzoru

$$F_\sigma(x, y) = \mathcal{N}_\sigma(x)\mathcal{N}_\sigma(y) = \frac{1}{\sigma\sqrt{2\pi}}\mathcal{N}_\sigma(r), \text{ gdzie } r = \sqrt{x^2 + y^2}.$$

Mając obraz oryginalny p , chcemy otrzymać obraz przefiltrowany q określony wzorem

$$q(\xi, \eta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(\xi - x, \eta - y)F_\sigma(x, y) dx dy.$$

Formalne określenie tego przekształcenia, zwanego **spłotem** funkcji p z F , wymaga rozszerzenia obrazów (funkcji określonych w prostokącie) na całą płaszczyznę; można uznać, że wartość funkcji p poza tym prostokątem jest zerem. Przetwarzając piksele, zastąpimy całkę kwadraturą.

335

Tensorowa definicja funkcji F_σ umożliwia wykonanie obliczenia w dwóch etapach; najpierw obliczymy

$$\hat{q}(\xi, \eta) = \sum_{i=-d}^d p(\xi - i, \eta)N_i \approx \int_{-\infty}^{\infty} p(\xi - x, \eta)\mathcal{N}_\sigma(x) dx,$$

a potem

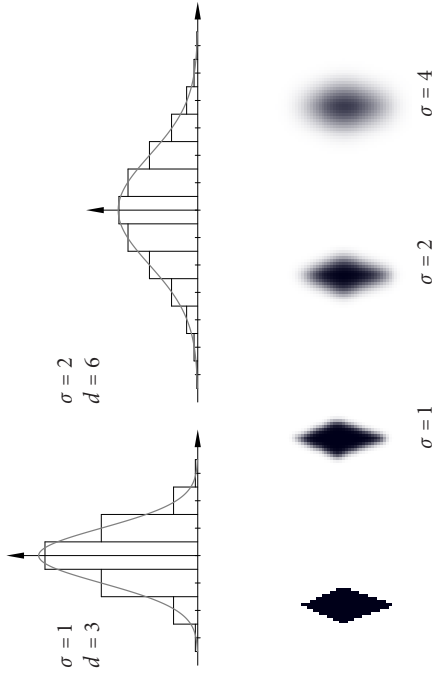
$$\hat{q}(\xi, \eta) = \sum_{j=-d}^d \hat{q}(\xi, \eta - j)N_j \approx q(\xi, \eta),$$

przy użyciu współczynników N_j będących wartościami średnimi funkcji \mathcal{N}_σ w przedziałach o długości 1:

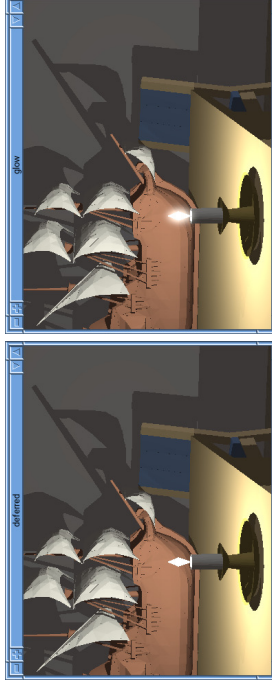
$$N_j = \int_{j-1/2}^{j+1/2} \mathcal{N}_\sigma(x) dx.$$

W każdym etapie wystarczy dla każdego piksela zsumować tylko $2d + 1$ składników.

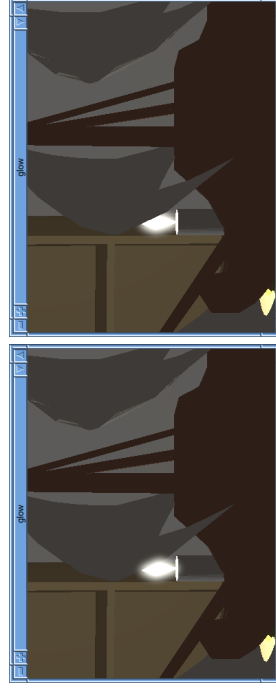
336



337



338



Filtrując obraz płomienia, można „przenieść” do sąsiednich pikseli głębokość punktu emitującego światło, po to, aby podczas wykonywania końcowego obrazu poddać poświacie testowi głębokości.

339

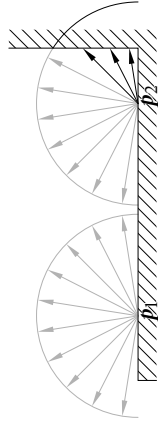
Najtrudniejsze jest w tej technice dobranie (obliczenie) mnożnika poświaty dodawanej do koloru piksela; musi on zależeć od wielkości obrazu źródła światła, a to oznacza, m.in. że najlepiej jest źródła światła narysować osobno, na tzw. billboardach i na nich dokonać filtrowania. Dla ustalenia mnożnika wypadłoby zliczyć piksele obrazu np. płomienia, przez osobny szader obliczeniowy.

Użycie billboardów pomaga też w sytuacji, gdy część lub całe źródło ma obraz na brzegu klatki lub tuż poza nim — wtedy poświata zanika, a nie powinna.

340

Modyfikowanie oświetlenia światłem rozproszonym

Znacznie prostszą alternatywą dla metody bilansu energetycznego jest technika zwana *screen space ambient occlusion* (SSAO). Polega ona na zmodyfikowaniu składnika opisującego światło rozproszone w otoczeniu, w zależności od kształtu otoczenia oświetlanego punktu. Jeśli są tam jakies obiekty, to zasłaniają część tego światła, ale jeśli same są oświetlone, to mogą odbijać część tego światła w stronę punktu widocznego w przetwarzanym pikselu.

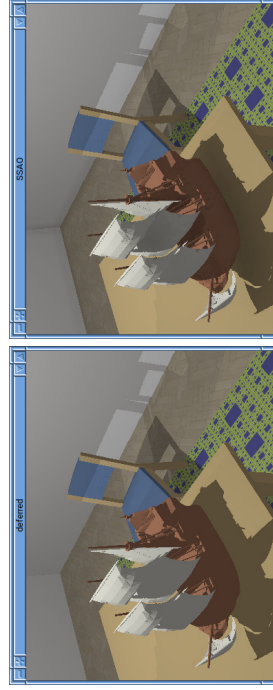


341

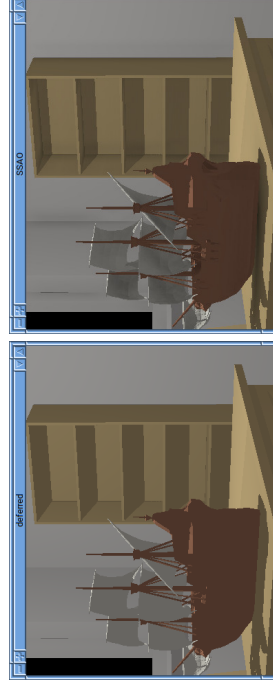
Pomysł polega na zbadaniu, czy w półkuli o promieniu r otrzymanej po przecięciu kuli płaszczyzną styczną do rysowanej powierzchni znajdują się jakieś obiekty. Część półsfery złożonej z kierunków, w których nie ma takich obiektów jest mnożnikiem intensywności światła rozproszonego w otoczeniu. Pozostała część może być mnożnikiem intensywności światła odbitego od pobliskich obiektów — dochodzącego bezpośrednio od punktowych źródeł światła.

W półsferze można wybrać pewną liczbę (np. rzędu sto kilkadziesiąt) wektorów i wykonać testowanie w kierunkach tych wektorów — posługując się zawartością G-bufora (w tym informacją o głębokości). Efekt jest taki, że na obrazie stają się widoczne kształty wnek i zakamarków, niosświetlonych bezpośrednio.

342



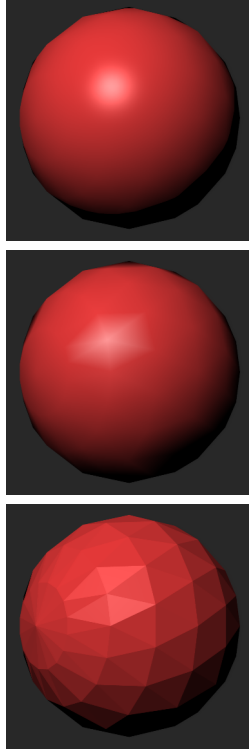
343



344

Metody cieniowania

Kolory pikseli obrazu trójkąta można obliczać na podstawie kolorów wierzchołków tego trójkąta — nazywa się to **cieniowaniem**. Różne metody cieniowania dają oczywiście różne wyniki.



345

W najprostszym przypadku można obliczyć kolor jednego punktu i wypełnić stałym kolorem cały trójkąt — to jest **cieniowanie płaskie**. Jeśli trójkąty mają być przybliżeniem fragmentu gładkiej powierzchni, to wyraźnie widać trójkąty.

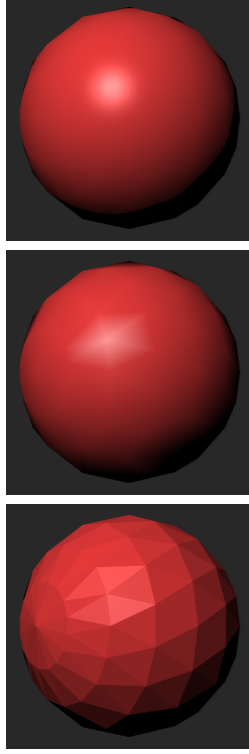
Metoda bardziej skomplikowana polega na liniowej interpolacji koloru między wierzchołkami — nazywa się to **cieniowaniem Gourauda** i działa dobrze dla powierzchni matowych, ale jeśli powierzchnia jest błyszcząca, to na gładkiej powierzchni zobaczymy kanciaste odbłaski. Stosując tę metodę, pewne odbłaski można też całkowicie zgubić.

Trzeci sposób to **metoda Phonga**; w niej dla każdego wierzchołka trójkąta podajemy wektor normalny przybliżanej przez trójkąty powierzchni (np. unormowany gradient funkcji, której warstwicą jest ta powierzchnia, albo iloczyn wektorowy pochodnych cząstkowych parametryzacji). Stosujemy interpolację liniową do wektorów normalnych w wierzchołkach trójkątów i tak otrzymane wektory normalne podstawiamy do modelu oświetlenia.

346

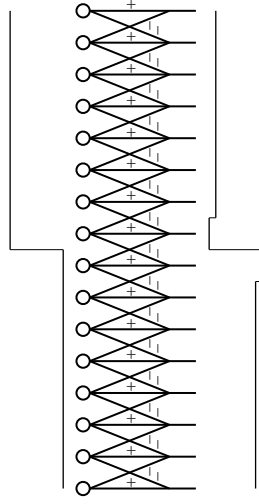
Metody cieniowania

Kolory pikseli obrazu trójkąta można obliczać na podstawie kolorów wierzchołków tego trójkąta — nazywa się to **cieniowaniem**. Różne metody cieniowania dają oczywiście różne wyniki.



345

Powód, dla którego krawędzie obszarów o różnych kolorach są dobrze widoczne wiąże się z działaniem zmysłu wzroku — proces wyodrębniania krawędzi zaczyna się w siatkówce oka, gdzie ma miejsce tzw. **hamowanie oboczne**.



Receptor, na który pada światło, wysyła sygnał pobudzający „jego” nerw i osłabiający sygnały przekazywane przez nerwy sąsiednich receptorów. Widoczny efekt — wzmocnienie krawędzi — nazywa się **efektem Macha**.

347

Od niewidocznych gołym okiem chropowatości zależy wygląd powierzchni, za którego otrzymanie odpowiada model oświetlenia. Obraz nierówności makroskopowych można otrzymać, zaburzając wektor normalny powierzchni podstawiany do modelu oświetlenia. Zaburzenia mogą być opisane przez tzw. **teksturę odkształceń**; po angielsku nazywa się to *bump mapping*.

Zobaczymy podstawy tej techniki dla powierzchni parametrycznych. Mamy regularną parametryzację $p: A \rightarrow \mathbb{R}^3$ klasy C^1 ; jednostkowy wektor normalny w punkcie $p(u, v)$ jest wartością **odwzorowania Gaussa**,

$$\mathbf{n}(u, v) = \frac{\mathbf{m}(u, v)}{\|\mathbf{m}(u, v)\|}, \quad \mathbf{m}(u, v) = \mathbf{p}_u(u, v) \wedge \mathbf{p}_v(u, v).$$

Dla uzyskania większej elastyczności wprowadzimy dodatkowe przekształcenie $q: A \rightarrow B \subset \mathbb{R}^2$. Funkcja $d: B \rightarrow \mathbb{R}$ opisuje odkształcenia powierzchni. Przy ich użyciu określamy parametryzację nowej powierzchni:

$$\hat{p}(u, v) = \mathbf{p}(u, v) + d(\mathbf{q}(u, v))\mathbf{n}(u, v).$$

348

Zakładamy, że funkcja d przyjmuje wystarczająco małe wartości bezwzględne. Na podstawie wzoru $\hat{p} = p + (d \circ q)n$ znajdziemy wektor normalny powierzchni opisanej przez tę parametryzację. Jej pochodne cząstkowe są kolumnami macierzy

$$D\hat{p} = Dp + D(d \circ q) \cdot n + (d \circ q)Dn \approx Dp + (Dd \cdot Dq)n.$$

Zakładamy, że wartości funkcji d są tak małe, że składnik $(d \circ q)Dn$, w którym są pochodne drugiego rzędu parametryzacji p , można pominąć. W końcowym wzorze mamy macierz Dd (gradient funkcji d o wymiarach 1×2) i macierz Dq o wymiarach 2×2 . Jeśli funkcja q jest przekształceniem afinicznym, to macierz Dq opisuje jego część liniową.

Jeśli funkcję d zadajemy jawnym wzorem, to trzeba wyprowadzić wzory opisujące jej pochodne cząstkowe. Można też reprezentować ją jako teksturę. Jeśli funkcja d jest reprezentowana za pomocą tablicy teksteli, to jej pochodne cząstkowe najprościej jest przybliżyć za pomocą różnic dzielonych, do czego może się przydać standardowa funkcja `textureOffset` dostępna w GLSL-u.

349

W aplikacji OpenGL-a szader fragmentów powinien otrzymać na wejściu parametry (u, v) odpowiadające przetworzanemu punktowi oraz pochodne cząstkowe parametryzacji p . Informacje na temat funkcji d i q mogą być przekazane w tekście lub w zmiennych jednolitych. Po znalezieniu macierzy $D\hat{p}$ trzeba obliczyć iloczyn wektorowy jej kolumn i unormować, otrzymując jednostkowy wektor normalny \hat{n} płata powierzchni o parametryzacji \hat{p} .

Do obliczeń oświetlenia podstawiamy wektor \hat{n} , ale jest tu pewien problem: oczywiście, ma on inny kierunek niż n . Do rozstrzygnięcia, którą stronę powierzchni widzi obserwator, wykorzystujemy wektor n . Jeśli iloczyn skalarny (v, n) i (v, \hat{n}) mają różne znaki, to do obliczeń oświetlenia trzeba użyć wektora n .

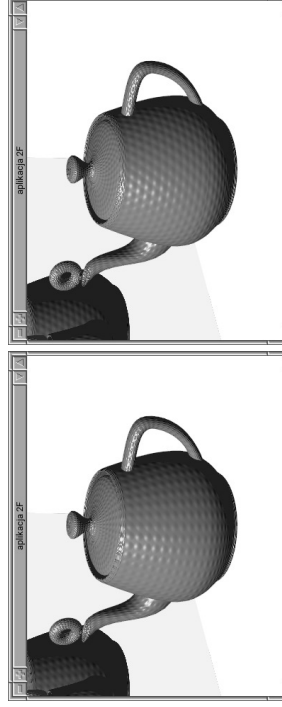
350

Jeśli wartości funkcji d są takie, że przemieszczenie punktów powierzchni na obrazie jest rzędu jednego lub paru pikseli (gdy obiekt jest oglądany w zbliżeniu), to może być potrzebna korekta tekstury odkształceń polegająca na modyfikacji współrzędnych tekstury.

Rozważmy płaty o parametryzacjach wymiernych $p(u, v)$ i $\hat{p}(u, v)$. Niech A oznacza macierz 4×4 przejścia do układu kostki standardowej — w przypadku rzutowania perspektywicznego to przejście nie jest przekształceniem afinicznym.

Niech Q, \bar{Q} i W oznaczają funkcje, których argumentem jest wektor w \mathbb{R}^4 . Pierwsza wybiera pierwsze 3 współrzędne, druga wybiera pierwsze 2 współrzędne, a ostatnia wybiera współrzedną wagową. Wtedy iloraz $r(P) = Q(P)/W(P)$ opisuje przejście od współrzędnych jednorodnych do kartezjańskich, a wektor $\bar{r}(P) = \bar{Q}(P)/W(P)$ składa się z pierwszych dwóch współrzędnych wektora $r(P)$.

352



351

Niech

$$\mathbf{P}(u, v) = \begin{bmatrix} \mathbf{p}(u, v) \\ 1 \end{bmatrix}, \quad \mathbf{N}(u, v) = \begin{bmatrix} \mathbf{n}(u, v) \\ 0 \end{bmatrix}, \quad \hat{\mathbf{P}}(u, v) = \begin{bmatrix} \hat{\mathbf{p}}(u, v) \\ 1 \end{bmatrix}.$$

Platy są opisane w układzie kostki standardowej przez parametryzację

$$\begin{aligned} \mathbf{r}(\mathbf{AP}(u, v)) &= \mathbf{Q}(\mathbf{AP}(u, v)) / W(\mathbf{AP}(u, v)), \\ \mathbf{r}(\mathbf{A}\hat{\mathbf{P}}(u, v)) &= \mathbf{Q}(\mathbf{A}\hat{\mathbf{P}}(u, v)) / W(\mathbf{A}\hat{\mathbf{P}}(u, v)). \end{aligned}$$

Funkcje $\mathbf{r}(\mathbf{AP}(u, v))$ i $\mathbf{r}(\mathbf{A}\hat{\mathbf{P}}(u, v))$ są parametryzacjami obrazów tych platów na ścianie kostki standardowej, która zostanie przekształcona na klatkę.

Niech (u_0, v_0) będzie punktem w dziedzinie plata, któremu odpowiada punkt przetwarzany przez szader fragmentów. Obraz punktu $\hat{\mathbf{p}}(u_0, v_0)$ ma na ścianie kostki standardowej współrzędne $(x, y) = \mathbf{r}(\mathbf{A}\hat{\mathbf{P}}(u_0, v_0))$. Chcemy znaleźć taki punkt (u^*, v^*) , aby było $(x, y) = \mathbf{r}(\mathbf{AP}(u^*, v^*))$. Do obliczenia współrzędnych wszelkich tekstur nakładanych na plakat zamiast (u_0, v_0) użyjemy punktu (u^*, v^*) . Tym samym uznamy, że przetwarzany fragment odpowiada punktowi $\hat{\mathbf{p}}(u^*, v^*)$.

353

Punkt (u^*, v^*) (który może nie istnieć) spełnia układ równań nieliniowych

$$\mathbf{r}(\mathbf{A}\hat{\mathbf{P}}(u_0, v_0)) = \mathbf{r}(\mathbf{AP}(u^*, v^*)).$$

Dla uproszczenia założymy, że $W(\mathbf{A}\hat{\mathbf{P}}(u_0, v_0)) \approx W(\mathbf{AP}(u^*, v^*))$, dzięki czemu otrzymamy układ równań $\mathbf{Q}(\mathbf{A}\hat{\mathbf{P}}(u_0, v_0)) = \mathbf{Q}(\mathbf{AP}(u^*, v^*))$. Mamy więc znaleźć miejsce zerowe funkcji

$$\mathbf{f}(u, v) = \mathbf{Q}(\mathbf{A}\hat{\mathbf{P}}(u_0, v_0) - \mathbf{P}(u, v)).$$

Użyjemy metody Newtona, ale wykonamy tylko jeden jej krok. Mamy

$$\mathbf{f}(u, v) = \mathbf{Q}\left(\mathbf{A}\left(\mathbf{P}(u_0, v_0) + d(\mathbf{q}(u_0, v_0))\mathbf{N}(u_0, v_0) - \mathbf{P}(u, v)\right)\right).$$

Zgodnie z definicją metody Newtona mamy obliczyć punkt

$$(u_1, v_1) = (u_0, v_0) - (\mathbf{D}\mathbf{f}(u_0, v_0))^{-1} \mathbf{f}(u_0, v_0).$$

354

Kolumny macierzy $\mathbf{D}\mathbf{f}(u, v)$ są pochodnymi cząstkowymi względem parametrów u, v . Ponieważ funkcja \mathbf{Q} jest przekształceniem liniowym, mamy

$$\mathbf{f}_u(u, v) = \mathbf{Q}(-\mathbf{A}\mathbf{P}_u(u, v)), \quad \mathbf{f}_v(u, v) = \mathbf{Q}(-\mathbf{A}\mathbf{P}_v(u, v)),$$

a zatem do wzoru opisującego krok metody Newtona możemy podstawić

$$\begin{aligned} \mathbf{f}(u_0, v_0) &= d(\mathbf{q}(u_0, v_0))\mathbf{Q}(\mathbf{A}\mathbf{N}(u_0, v_0)), \\ \mathbf{D}\mathbf{f}(u_0, v_0) &= \begin{bmatrix} -\mathbf{Q}(\mathbf{A}\mathbf{P}_u(u_0, v_0)), & -\mathbf{Q}(\mathbf{A}\mathbf{P}_v(u_0, v_0)) \end{bmatrix}. \end{aligned}$$

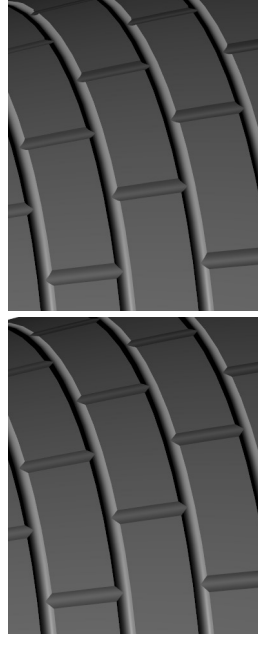
Kolejne kroki metody Newtona wymagałyby obliczenia punktu $\mathbf{p}(u_1, v_1)$ oraz pochodnych parametryzacji plata \mathbf{p} w punkcie (u_1, v_1) i dalszych i to musiałby robić szader fragmentów. Ale pochodne w punkcie (u_0, v_0) szader fragmentów może otrzymać od szadera rozdrabniania.

355

Mając punkt (u_1, v_1) , można obliczyć gradient funkcji $d(\mathbf{q}(u, v))$ w tym punkcie, po czym do wzoru

$$\mathbf{D}\hat{\mathbf{p}} = \mathbf{D}\mathbf{p} + (\mathbf{D}d \cdot \mathbf{D}\mathbf{q})\mathbf{n}$$

zamiast pochodnych i wektora normalnego plata \mathbf{p} w punkcie (u_1, v_1) możemy podstawić pochodne i wektor normalny w punkcie (u_0, v_0) . Zobaczymy efekt:



Z lewej strony jest obrazek bez korekty, a z prawej z opisaną korektą.

356

Uwagi: Opisany sposób nie zawsze poprawia obrazek — pokazana tekstura odkształceń (funkcja d) ma nieciągłe pochodne cząstkowe, ale w punktach nieciągłości jest $d = 0$. Dzięki temu efekt jest w tym przykłady zadowalający.

Poszukiwany punkt (u^*, v^*) może nie istnieć.

Jeśli punkt (u^*, v^*) istnieje, to punkt (u_1, v_1) jest tylko jego przybliżeniem.

Jeśli punkt (u_1, v_1) leży poza dziedziną płata, to można zrezygnować z korekty, zastąpić ten punkt najbliższym punktem w dziedzinie płata albo dokonać okresowego rozszerzenia tekstury.

Nie ma jednej, „jedynie słusznej” metody postępowania.

Opisany sposób nie może zmienić sylwetki rysowanego obiektu.

357

Metoda śledzenia promieni

Metoda śledzenie promieni (*ray tracing*) powstała na przełomie lat 70-tych i 80-tych XX wieku. Jej celem jest umożliwienie otrzymania na obrazach efektów wynikających z wielokrotnych odbić i załamania światła na powierzchniach lustrzanych i na gładkich granicach przezroczystych ośrodków, takich jak szkło, woda itd. Podstawowy algorytm w niewielkim stopniu uwzględnia rozproszone odbicia światła prowadzące do powstawania półcieni, ale techniki opracowane w związku z tym algorytmem są wykorzystywane nawet w numerycznych metodach rozwiązywania równania bilansu energetycznego dla scen składających się z powierzchni nielambertowskich.

358

W klasycznym algorytmie śledzenia promieni otwiera się drogi, jakie przebyły fotony, które dotarły do obserwatora — jest to więc śledzenie promieni *wstecz*. Promień jest półprostą, wyznaczoną przez punkt początkowy i wektor kierunkowy (mający określony zwrot).

Położenie obserwatora i klatka są opisane w układzie współrzędnych świata.

Wybrane punkty klatki (np. odpowiadające środkom pikseli) wyznaczają **promienie pierwotne**. Ich początkiem jest położenie obserwatora, a wektor kierunkowy każdego z nich jest różnicą punktu klatki i początku.

Zadanie pomocnicze, o podstawowym znaczeniu, polega na znalezieniu przecięcia promienia z powierzchnią obiektu sceny, przy czym ma to być punkt przecięcia położony najbliżej początku promienia. Z tego punktu wychodzą **promienie wtórne**, których przecięcia z obiektami trzeba znaleźć tak samo. W ten sposób powstają promienie wtórne drugiego, trzeciego i dalszych rzędów.

359

Są trzy rodzaje promieni wtórnych:

- **promienie do źródeł światła** — jeśli na odcinku promienia od początku do źródła światła nie ma przecięć z innymi obiektami, to początek promienia jest oświetlony, w przeciwnym razie jest w cieniu,
- **promienie odbite** — jeśli powierzchnia, na której leży początek promienia jest lustrem, to jego kierunek wyznacza się na podstawie kierunku promienia niższego rzędu i wektora normalnego powierzchni. Jeśli jest matowa, to można wysłać kilka lub kilkanaście promieni odbitych w losowych kierunkach, choć w podstawowym wariancie się tego nie robi,
- **promienie załamane** — jeśli powierzchnia jest granicą przezroczystych ośrodków o różnych współczynnikach załamania światła. Współczynnik załamania światła szkła lub wody zmienia się z długością fali, co może spowodować potrzebę wygenerowania kilku promieni załamanych, aby otrzymać efekt rozszczepienia światła białego przez pryzmat.

360

Promień pierwotny jest korzeniem **drzewa promieni**, które generuje się na bieżąco i przeszukuje metodą DFS. Obliczenie koloru piksela (lub subpiksela) odbywa się podczas powrotu z rekurencyjnych procedur śledzenia promieni wtórnych — intensywność światła niesionego przez te promienie jest, na podstawie lokalnego modelu oświetlenia, używana do obliczenia intensywności światła niesionego przez promień niższego rzędu. Dla powierzchni matowych (lub niedoskonałe lustrzanych) uwzględnia się też (jakoś przyjęta) intensywność światła rozproszonego w otoczeniu początku promienia.

361

Dla zapewnienia własności stopu algorytmu ogranicza się rząd promieni wtórnych, czyli głębokość rekurencji (albo wysokość drzewa promieni).

Przeszukiwanie drzewa promieni można zakończyć na niższym poziomie. Każdy promień ma swoją **wagę**, która określa, jaki wpływ światło niesione przez ten promień ma na kolor piksela na końcowym obrazie. Promienie pierwotne mają wagę równą 1. Wagi promieni wtórnych są obliczane na podstawie lokalnego modelu oświetlenia w ich punktach początkowych — odpowiedni ułamek jest mnożony przez wagę promienia niższego rzędu. Promienie, których wagi są mniejsze niż przyjęta wartość progowa, pomijają się.

362

Ulepszeniem klasycznego śledzenia promieni wstecz jest algorytm **dwukierunkowego śledzenia promieni**, bardziej znany jako **algorytm map fotonowych** (ang. *photon mapping*). Jest to algorytm dwuetapowy. W pierwszym etapie generowane są promienie pierwotne o początkach w punktowych źródłach światła i o losowych kierunkach — według rozkładu zgodnego z charakterystyką źródła światła (np. reflektory wysyłają światło tylko w obrębie pewnego stożka). Wzdłuż tych promieni są wysyłane **fotony**, czyli paczki energii, niosące określony strumień energetyczny. Fotony trafiają w punkty przecięcia promieni z powierzchniami sceny, skąd są dalej rozsyłane wzdłuż promieni wtórnych. Strumień energetyczny fotonu jest rozdzielany między promienie wtórne, przy czym część tej energii zanika. W tych obliczeniach ma zastosowanie dwukierunkowa funkcja odbicia i załamania światła.

Dla każdej powierzchni trzeba utworzyć **mapę fotonową**, w której są przechowywane informacje o punktach trafionych przez fotony i o ich energiach. Często stosowane są tu k -d drzewa. Podstawową informacją możliwą do uzyskania z mapy fotonowej jest wykaz fotonów, które trafiły w powierzchnię w pobliżu (tj. nie dalej niż w ustalonej odległości od) dowolnego punktu.

363

W drugim etapie wykonuje się klasyczne śledzenie promieni wstecz, ale w obliczeniach oświetlenia uwzględnia się światło przyniesione do odpowiednich kawałków powierzchni (w otoczeniu punktów przecięć) przez fotony. Na podstawie mapy fotonowej można obliczyć (w przybliżeniu) irradancję światła padającego na powierzchnię w otoczeniu punktu przecięcia z promieniem i dalej obliczyć radiancję światła odbitego w kierunku początku promienia. Dla powierzchni lustrzanych można generować tylko wtórne promienie odbite, a dla powierzchni matowych pewną liczbę promieni o losowych kierunkach.

Algorytm map fotonowych umożliwia otrzymanie półcieieni, a także **kaustyk**, czyli efektów skupienia światła przez soczewki i inne przedmioty szklane lub np. wodę w szklance, oraz fale na powierzchni wody i zakrzywione lustro. Można w tym algorytmie przyjąć mniejsze wysokości drzew promieni, co oczywiście przekłada się na czas obliczeń.

364

Śledzenie promieni ma dwa najważniejsze elementy: znajdowanie przecięć promieni z obiektami i stosowanie lokalnych modeli oświetlenia. Największy koszt znajdowania przecięć, z uwagi na złożoność zadania geometrii obliczeniowej, jakim jest znalezienie właściwego obiektu, z którym przecina się promień (scena może składać się z setek obiektów, z których każdy może składać się z setek lub tysięcy pryzmatów takich jak trójkąty).

Dodatkowo powierzchnie obiektów mogą być zakrzywione — pewne implementacje zastępują takie powierzchnie przybliżającymi je trójkątami, inne rozwiązują równania nieliniowe, co zwiększa dokładność obrazów kosztem skomplikowania algorytmu i zwiększenia czasu obliczeń.

365

Powierzchnia może być opisana w postaci niejawniej (jako zbiór miejsc zerowych funkcji trzech zmiennych) lub w postaci parametrycznej. Założymy, że promień jest reprezentowany przez punkt początkowy $\mathbf{a} = (x_a, y_a, z_a)$ i wektor kierunkowy $\mathbf{w} = (x_w, y_w, z_w)$, a zatem składa się z punktów $\mathbf{a} + t\mathbf{w}$ dla $t > 0$.

Powierzchnia zadana niejawnie jest zbiorem miejsc zerowych funkcji $F: \mathbb{R}^3 \rightarrow \mathbb{R}$, czyli punktów \mathbf{p} , takich że $F(\mathbf{p}) = 0$. Po podstawieniu parametryzacji promienia powstaje równanie z jedną niewiadomą:

$$F(\mathbf{a} + t\mathbf{w}) = 0.$$

Jeśli funkcja F nie jest wielomianem pierwszego stopnia, to to równanie jest nieliniowe.

Dla **powierzchni parametrycznej** o parametryzacji $\mathbf{p}: A \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$ trzeba rozwiązać układ trzech równań

$$\mathbf{p}(u, v) - \mathbf{a} - t\mathbf{w} = \mathbf{0},$$

z niewiadomymi $(u, v) \in A$ oraz $t > 0$. Ze względu na zmienną t te równania są liniowe, co można i warto wykorzystać w algorytmie rozwiązywania układu.

366

Przykłady: Płaszczyzna przechodząca przez punkt \mathbf{p}_0 i mająca wektor normalny \mathbf{n} jest zbiorem punktów \mathbf{p} spełniających równanie

$$\langle \mathbf{n}, \mathbf{p} - \mathbf{p}_0 \rangle = 0.$$

Po wstawieniu parametryzacji promienia mamy równanie

$$\langle \mathbf{n}, \mathbf{a} + t\mathbf{w} - \mathbf{p}_0 \rangle = 0.$$

Jeśli $\langle \mathbf{n}, \mathbf{w} \rangle \neq 0$, to jego rozwiązaniem jest liczba

$$t = \frac{\langle \mathbf{n}, \mathbf{p}_0 - \mathbf{a} \rangle}{\langle \mathbf{n}, \mathbf{w} \rangle}.$$

Wystarczy sprawdzić, czy $t > 0$ i jeśli tak, obliczyć punkt przecięcia $\mathbf{p} = \mathbf{a} + t\mathbf{w}$.

367

Znaleźć punkt przecięcia promienia z trójkątem o wierzchołkach $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ można tak: niech $\mathbf{v}_1 = \mathbf{p}_1 - \mathbf{p}_0, \mathbf{v}_2 = \mathbf{p}_2 - \mathbf{p}_0$. Trójkąt składa się z punktów $\mathbf{p} = \mathbf{p}_0 + u_1\mathbf{v}_1 + u_2\mathbf{v}_2$, dla których $u_1, u_2 \geq 0, u_1 + u_2 \leq 1$. Wektorem normalnym płaszczyzny trójkąta jest wektor $\mathbf{n} = \mathbf{v}_1 \wedge \mathbf{v}_2$. Podstawiając promień do równania płaszczyzny i rozwiązując ze względu na t , dostaniemy jak poprzednio

$$t = \frac{\langle \mathbf{n}, \mathbf{p}_0 - \mathbf{a} \rangle}{\langle \mathbf{n}, \mathbf{w} \rangle}.$$

Jeśli $t > 0$, to dalej mamy układ trzech równań liniowych z dwiema niewiadomymi,

$$\mathbf{p} = \mathbf{p}_0 + u_1\mathbf{v}_1 + u_2\mathbf{v}_2,$$

przy czym jest to układ niesprzeczny. Możemy go przedstawić w postaci $A\mathbf{x} = \mathbf{b}$, z macierzą $A = [\mathbf{v}_1, \mathbf{v}_2]$, wektorem $\mathbf{b} = \mathbf{a} + t\mathbf{w} - \mathbf{p}_0$ i wektorem niewiadomych $\mathbf{x} = [u_1, u_2]^T$. Układ rozwiążemy jak liniowe zadanie najmniejszych kwadratów, za pomocą **pseudoodwrótności** macierzy $A: A^+ = (A^T A)^{-1} A^T$. Obliczamy

$$\mathbf{x} = A^+ \mathbf{b},$$

a potem sprawdzamy, czy $u_1, u_2 \geq 0, u_1 + u_2 \leq 1$.

368

Oczywiście, ten sam wynik (z dokładnością do błędów zaokrągleń) można by otrzymać, rozwiązując układ równań z macierzą 3×3

$$[v_1, v_2, -w] \begin{bmatrix} u_1 \\ u_2 \\ t \end{bmatrix} = a - p_0,$$

albo układ z macierzą 4×4 :

$$\begin{bmatrix} p_0 & p_1 & p_2 & -w \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ t \end{bmatrix} = \begin{bmatrix} a \\ a \\ 1 \\ 1 \end{bmatrix}.$$

Wcześniej podany sposób umożliwia wykonanie pewnych obliczeń w preprocesingu — wektor normalny n i macierz A^+ zależą tylko od wierzchołków trójkąta.

Dla każdego trójkąta można je obliczyć tylko raz i zapamiętać, a potem dla setek promieni, których przecięcia z trójkątem wyznaczamy, wykonywać tylko mnożenia odpowiednich wektorów przez macierz A^+ o wymiarach 2×3 . Kosztem jest zajęcie dodatkowej pamięci przez wyniki preprocesingu (9 liczb na każdy trójkąt).

369

Sfera o środku $c = (x_c, y_c, z_c)$ i promieniu r jest zbiorem miejsc zerowych funkcji

$$F(x, y, z) = (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2.$$

Po podstawieniu parametryzacji promienia otrzymamy równanie kwadratowe $at^2 + 2bt + c = 0$, o współczynnikach

$$\begin{aligned} a &= x_w^2 + y_w^2 + z_w^2, \\ b &= x_w(x_a - x_c) + y_w(y_a - y_c) + z_w(z_a - z_c), \\ c &= (x_a - x_c)^2 + (y_a - y_c)^2 + (z_a - z_c)^2 - r^2. \end{aligned}$$

Jeśli wektor kierunkowy promienia jest jednostkowy, to $a = 1$, zatem warto wektor każdego promienia natychmiast po wygenerowaniu unormować. Dalej możemy obliczyć $\Delta = b^2 - c$ i jeśli $\Delta \geq 0$, to obliczamy $t_1 = -b - \sqrt{\Delta}$, a jeśli $t_1 \leq 0$, to obliczamy $t_2 = -b + \sqrt{\Delta}$. Przed sprawdzeniem, czy równanie ma pierwiastki rzeczywiste, trzeba wykonać 7 mnożeń i 9 dodawań (r^2 dla sfery możemy obliczyć w preprocesingu), a potem obliczyć 1 pierwiastek i wykonać 2 odejmowania.

370

Znalezienie (wszystkich) punktów przecięcia promienia z **platem Béziera** wymaga rozwiązania układu równań

$$\sum_{i=0}^n P_{ij} B_i^n(u) B_j^n(v) - a - tw = 0.$$

Możemy w tym celu przejść do takiego układu współrzędnych, którego początkiem jest punkt a , a wektor kierunkowy promienia jest wersorem osi z . Możemy w tym celu dokonać przesunięcia, tj. obliczyć wektory (punkty kontrolne) $\hat{P}_{ij} = P_{ij} - a$, a potem dokonać odbicia symetrycznego (Householdera) przeprowadzającego wektor w na wektor $\pm e_3 = (0, 0, \pm 1)$. Wektory $\tilde{P}_{ij} \in \mathbb{R}^2$, składające się z pierwszych dwóch współrzędnych obrazów punktów P_{ij} w tym przekształceniu są punktami kontrolnymi płaskiego plata Béziera, $\tilde{P}(u, v)$. Trzeba znaleźć miejsca zerowe tej parametryzacji, tj. punkty $(u, v) \in [0, 1]^2$, takie że $\tilde{P}(u, v) = 0$.

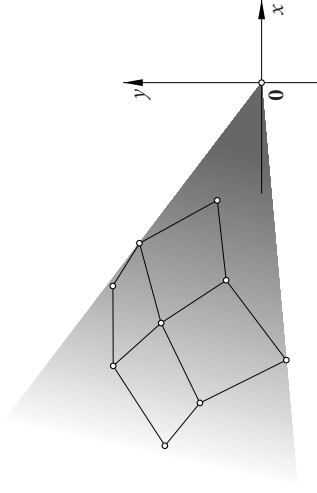
371

Na podstawie własności otoczki wypukłej platów Béziera, jeśli otoczka wypukła zbioru punktów \tilde{P}_{ij} nie zawiera punktu $0 = (0, 0)$, to układ równań nie ma rozwiązań. Dalej, jeśli ten **test otoczki wypukłej** dopuszcza istnienie rozwiązań, to można sprawdzić, czy rozwiązanie może być tylko jedno. W tym celu trzeba zbadać, czy dwie rodziny wektorów, $\{\Delta_1 \tilde{P}_{ij} = \tilde{P}_{i+1,j} - \tilde{P}_{ij}\}_{i,j}$ oraz $\{\Delta_2 \tilde{P}_{ij} = \tilde{P}_{i,j+1} - \tilde{P}_{ij}\}_{i,j}$ są zawarte w dwóch rozłącznych stożkach wypukłych zawartych w stożku wypukłym (dowód odpowiedniego twierdzenia pomiję). Jeśli tak, to rozwiązanie jest w kwadracie $[0, 1]^2$ jednoznaczne. W przeciwnym razie można dokonać podziału plata na dwie części algorytmem de Casteljau, w odpowiednich prostokątach otrzymanych z podziału dziedziny wprowadzić lokalne zmienne przyjmujące wartości od 0 do 1 i stosować testy otoczki wypukłej i jednoznaczności rozwiązania do części plata.

Jeśli fragment plata przeszedł test jednoznaczności, to można użyć metody Newtona. Jeśli znajdzie ona rozwiązanie w kwadracie $[0, 1]^2$, to wiadomo, że innych rozwiązań tam nie ma. Jeśli zawiedzie lub znajdzie rozwiązanie poza tym kwadratem, to pląt dzielimy na kawałki dalej.

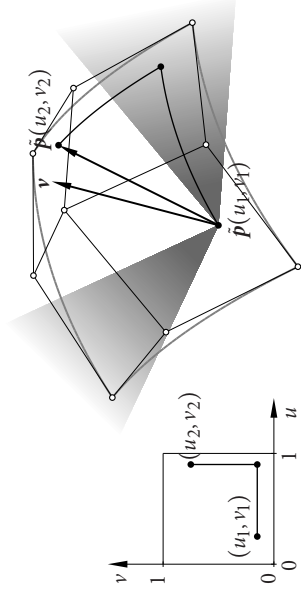
372

Test otoczki wypukłej:



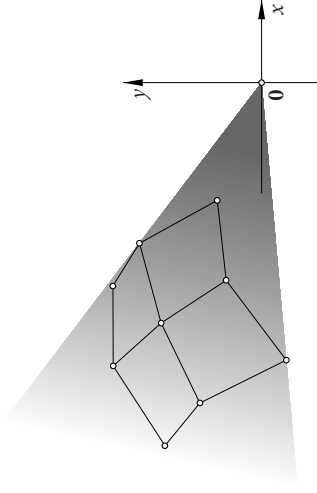
373

Test jednoznaczności rozwiązania; rysunek jest szkicem dowodu:



374

Test otoczki wypukłej:



373

Śledzenie promieni i konstrukcyjna geometria brył

Obiekt może być określony jako bryła CSG i reprezentowany przez drzewo CSG, w którego liściach są prymitywy. Punkt przecięcia promienia z taką bryłą, który należy znaleźć, jest punktem przecięcia promienia z powierzchnią pewnego prymitywu.

Przecięcia promienia z bryłami składają się z odcinków. Zadanie znalezienia przecięcia promienia z trójwymiarową bryłą CSG można zatem sprowadzić do zadania jednowymiarowego — operacje mnogociowe przeprowadzamy na częściach wspólnych promienia z prymitywami, tj. na zbiorach odcinków.

Identyfikator prymitywu powinien być atrybutem każdego końca tych odcinków, dzięki czemu po znalezieniu punktu przecięcia promienia z bryłą CSG położonego najbliżej początku promienia można użyć atrybutów powierzchni właściwego prymitywu.

376

Implementacja metody znajdowania przecięć może się posługiwać stosem, na którym umieszczamy na początku cały płat, tj. tablicę punktów \vec{P}_{ij} oraz dziedzinę, tj. kwadrat $[0, 1]^2$. W pętli, do momentu opróżnienia stosu, zdejmujemy element i wykonujemy test otoczki wypukłej. Jeśli nie wyklucza istnienia rozwiązania, to wykonujemy test jednoznaczności rozwiązania. Jeśli nie wykluczył istnienia dwóch lub więcej rozwiązań, to dzielimy płat na dwie części algorytmem de Casteljau i pakujemy (z opisem prostokątów będących ich dziedzinami) na stos. Jeśli test jednoznaczności przeszedł, to próbujemy rozwiązać układ równań metodą Newtona. Jeśli metoda znalazła rozwiązanie w dziedzinie fragmentu płata, to obliczamy punkt przecięcia. Jeśli metoda zawiodła, to dzielimy płat i wstawiamy kawałki na stos.

W ten sposób metoda szybka, ale zawodna (Newtona), została uzupełniona metodą wolniejszą, ale niezawodną.

375

Techniki przyspieszające

Podstawowym problemem do rozwiązania w implementacji śledzenia promieni jest zmniejszanie kosztu obliczeń przecięć promieni z obiektami. Obiektów w scenie jest dużo (setki lub tysiące), a promieni jest jeszcze więcej (miliony), przy czym większość par (promień, obiekt) nie ma przecięć.

Szybkie wykrywanie braku przecięć dla danej pary (jeśli ich nie ma) może przyspieszyć obliczenia o czynnik stały. Istotne zmniejszenie złożoności obliczeniowej można uzyskać, jeśli się będzie eliminować większość par (promień, obiekt) bez ich sprawdzania.

377

Do szybkiej eliminacji par (promień, obiekt) służą bryły otaczające (*bounding volumes*). Jeśli obiekt leży w pewnej bryle, z którą promień się nie przecina, to z tym obiektem też nie ma przecięć. Wybiera się takie bryły otaczające, które

- dają możliwość szybkiego sprawdzenia, czy promień jest z nimi rozłączny,
- możliwie ciasno otaczają objekty.

Najczęściej stosowane są kule i kostki prostopadłościowe (*axis aligned bounding boxes, AABB*). Sprawdzenie, czy promień przecina się z kulą lub kostką jest tanie, dopiero po wykryciu przecięcia można przystąpić do obliczania punktów przecięcia z pełną dokładnością za pomocą bardziej kosztownej procedury numerycznej. Wykonujemy ją na przykład dla płatów Béziera, których cała siatka kontrolna zawiera się w kuli lub kostce.

378

Jeśli obiekt ma taki kształt, że zawierająca go kula lub kostka obejmuje dużo dodatkowego miejsca (np. obiekt jest wydłużony w kierunku nierównoległym do żadnej osi układu), to wstępny test przejdzie dużo promieni rozłącznych z obiektem. W takich przypadkach można wprowadzić kilka brył otaczających (np. kulę i walec lub kostki o krawędziach równoległych do osi różnych układów współrzędnych) i pełny koszt obliczeń numerycznych znajdowania przecięć ponosić, gdy promień przecina wszystkie te bryły.

Aby zmniejszyć rząd złożoności obliczeniowej, trzeba wprowadzić pewną hierarchię przestrzenną obiektów w scenie. Najczęściej to jest pewne drzewo, którego korzeń reprezentuje całą scenę, a poszczególne poddrzewa reprezentują części tej sceny zajmujące różne miejsca w przestrzeni. W wierzchołkach drzewa będą pojedyncze prymitywy lub krótkie wykazy prymitywów. Każdy wierzchołek drzewa ma bryłę otaczającą. Mając promień, przeszukujemy drzewo, pomijając poddrzewa, z których bryłami otaczającymi promień się nie przecina.

379

Do budowy drzewa można podchodzić na dwa sposoby: pierwszy to wykorzystanie naturalnej hierarchii obiektów w scenie (np. traktowanie kół, drzwi i karoserii jak jednego obiektu złożonego, tj. samochodu reprezentowanego przez wierzchołek wewnętrzny drzewa). Bryły otaczające dla wierzchołków znajdujemy w kolejności od liści do korzenia.

Drugi sposób, to zbudowanie drzewa ósemkowego lub *k-d* drzewa, dostosowując podział przestrzeni adaptacyjnie do rozmieszczenia obiektów. Mając listę wszystkich obiektów, otaczamy je kostką, którą następnie dzielimy. Podział kostki umożliwia przeniesienie obiektów mieszczących się w jej częściach do list obiektów w tych częściach.

380

Mając promień, przeszukujemy drzewo w celu wykrycia jego przecięć z obiektami obecnymi w liściach tych wierzchołków, z których kostkami promieni się przecina. Kolejność przeszukiwania możemy wybrać tak, aby najpierw zbadać kostki, których przecięcie z promieniem jest najbliższe początku promienia — znalezienie w takiej kostce punktu przecięcia z obiektem zwalnia z potrzeby przeszukiwania dalszych poddrzew (chyba, że obiekt jest bryłą CSG).

Dla danego promienia może być potrzebne przeszukanie wielu gałęzi drzewa, zanim zostanie znaleziony właściwy obiekt. Drzewa ośminkowe mają tę zaletę, że duże puste obszary w przestrzeni zawierają stosunkowo duże kostki, w związku z czym, w poszukiwaniu obiektu przeciętego przez promień szybko się je przebywa. Ale koszt przeszukiwania drzewa od korzenia do liści jest proporcjonalny do odległości liścia od korzenia.

381

Alternatywnym rozwiązaniem jest dokonanie podziału kostki zawierającej scenę na jednakowe **woksele**, ustawione w prostopadłościenną trójwymiarową tablicę. Wielkość wokseli jest na tyle mała, by w każdym z nich było zawarte niewiele obiektów, ale na tyle duża, by stosunkowo niewiele obiektów przecinało się z więcej niż jednym wokselem.

Zaletą tej metody (którą wynalazł ok. 1985 r. Akira Fujimoto) jest szybkość dostępu do woksela zawierającego dowolny punkt kostki — odbywa się to w czasie stałym. Jej wadą jest brak adaptacji. Duże puste obszary są podzielone na woksele o takiej samej wielkości jak obszary, w których są obiekty, a żeby przebyć taki obszar wzdłuż promienia, trzeba przejść przez wszystkie puste woksele.

382

Najprostsze wykorzystanie sprzętu do przyspieszania śledzenia promieni polega na użyciu G-bufora: umożliwia on „obsłużenie” wszystkich promieni pierwotnych. Wykonujemy obraz przy użyciu algorytmu widoczności z buforem głębokości, w pozaekranowym buforze ramki. Załączniki obrazu tego bufora, zamiast kolorów, które można przypisać pikselom na ekranie, mają pomieścić inne informacje. Najważniejsza z nich to numer (lub inny identyfikator) obiektu widocznego w danym pikselu. Inny atrybut, który warto zapamiętać, to wektor normalny, mogą być też parametry tekstury itd.

Pozaekranowy bufor ramki może mieć kilka załączników obrazu (`GL_COLOR_ATTACHMENTi`), tj. tekstur lub buforów roboczych (*renderbuffers*) z tablic pikseli różnych typów — zmiennopozycyjnych lub stałopozycyjnych (specyfikacja przewiduje, że może ich być nawet 32, choć implementacje zazwyczaj pozwalają na mniej). Szader fragmentów może wyprowadzić do każdego z nich odpowiedni wynik.

383

Po narysowaniu sceny w G-buforze następuje druga faza — generowania i przetwarzania promieni wtórnych. Na podstawie współrzędnych piksela w klatce, liczby zapamiętanej w buforze głębokości i macierzy przejścia od układu świata do układu kostki standardowej, możliwe jest obliczenie współrzędnych w układzie świata punktu na powierzchni, który jest początkiem promienia, a także wektora kierunkowego promienia pierwotnego dla tego piksela.

Mając wektor normalny i identyfikator obiektu (dla którego znamy własności optyczne powierzchni), możemy wygenerować promienie wtórne pierwszego rzędu. Dla nich i dla promieni wtórnych wyższych rzędów rekurencyjne śledzenie promieni odbywa się dostępnymi środkami. To jest trudne zadanie, bo GPU wymaga **jednolitości obliczeń** — wszystkie rdzenie wykonujące wątki obliczeniowe w grupie roboczej albo wykonują tę samą instrukcję, albo czekają. Niedozwolone jest także wywoływanie rekurencyjne podprogramów.

384

Brak rekurencji można ominąć za pomocą jawnie zrealizowanego stosu, ale pozostaje zadanie wyznaczenia przecięć promieni z obiektami przez wątki szadera przetwarzającego piksele obrazu w G-buforze. Dla sceny, przed rysowaniem (w preprocessingu), trzeba utworzyć jakąś strukturę danych (np. drzewo ósemkowe), którego wierzchołki dają dostęp do odpowiednich list obiektów (np. trójkątów).

Sprzęt rozpowszechniony obecnie nie ma elementów ułatwiających realizację tego zadania, choć to właśnie teraz się zmienia. Topowe procesory graficzne firmy NVIDIA są wyposażone w podukłady przeznaczone do szybkiego znajdowania przecięć z trójkątami. Niestety, korzystanie z tych możliwości nowego sprzętu nie jest (i nie ma być) dostępne w aplikacjach OpenGL-a — firma opracowała tylko odpowiednie rozszerzenia standardów DirectX i Vulkan. Ale zadanie daje się rozwiązywać środkami dostępnymi w OpenGL-u, co może działać tylko kilka razy wolniej, a przy tym będzie działać też na sprzęcie wcześniejszej generacji i na sprzęcie innych producentów.

385

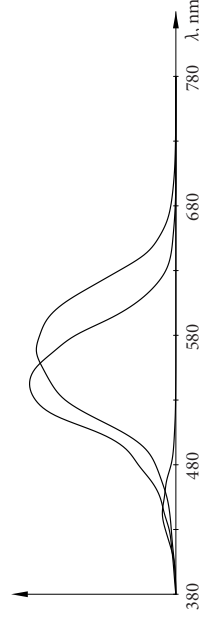
Można znaleźć w sieci materiały na temat sposobu zaprogramowania śledzenia promieni w aplikacji Vulkana, przy czym opis nie jest specjalnie precyzyjny — trudno jest znaleźć opis stosowanej struktury danych, która umożliwia wyszukiwanie właściwego trójkąta dla ustalonego promienia. Istotnym jej elementem są kostki otaczające (AABB), ale poza tym struktura ta jest ukryta przed programistą, który ma spowodować jej zbudowanie, przy czym określa tylko dwa poziomy tej struktury:

- BLAS (*bottom level acceleration structure*) — to są pojedyncze obiekty (zestawy niewielu trójkątów), które znajdują się w takich strukturach. Cała scena składa się z tak podanych obiektów, przy czym mogą one być powielane w scenie.
- TLAS (*top level acceleration structure*) — „cała” struktura do znajdowania przecięć.

386

Współrzędne w przestrzeni barw

Receptory w siatkówce oka ludzkiego są dwóch rodzajów, tzw. **czopki i pręciki**. Pręciki są bardziej czułe, ale nie mogą różnicować kolorów. Czopki są trzech rodzajów, które wykazują największą czułość dla fal elektromagnetycznych (światła) o różnych długościach. Zatem wrażenia barwne przekazywane przez oko są trójwymiarowe.



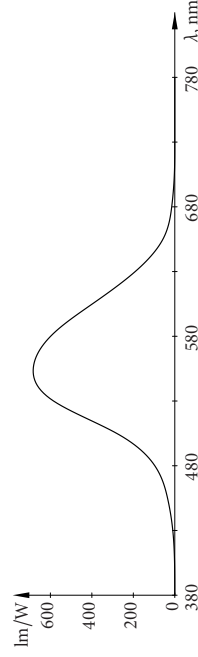
388

Struktury przyspieszające śledzenie promieni można zbudować od zera (*build*) lub zmodyfikować (*update*), co może trwać znacznie krócej, ale dopuszczalne są tylko niektóre zmiany. Aplikacja powinna dostarczyć szadery wywoływane po znalezieniu przecięcia promienia z trójkątem i wywoływane po niezalezieniu trójkąta przeciętego przez promień, ale opis w podręczniku NVIDIA jest moim zdaniem celowo niejasny; zmusza do wcześniejszego dogłębnego poznania standardu Vulkan i starannie pomija najciekawsze fragmenty implementacji.

No, cóż, jest co studiować.

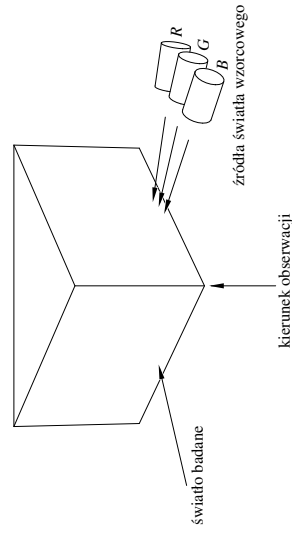
387

Sumaryczna czułość receptorów w zależności od długości fali jest opisana przez funkcję zwaną **skutecznością świetlną** (*luminous efficacy*) — opisuje ona przelicznik mocy światła (mierzonej w watach) na strumień świetlny (w lumenach). Jej maksymalna wartość dla $\lambda = 555$ nm to 683 lm/W. Dla światła białego, w zależności od temperatury światła jest to od ok. 50 lm/W do 250 lm/W.



389

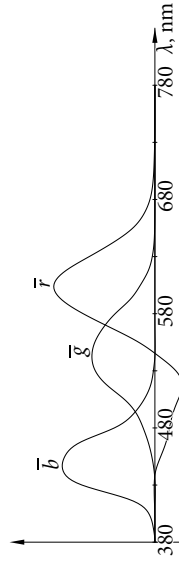
Do badania postrzegania barw stosuje się **kolorymetr klinowy**. Biały klin jest umieszczony w komorze o czarnych ścianach; jedną ścianę klina oświetla światło badane, a drugą źródła światła wzorcowego — żarówka z filtrem przepuszczającym światło o długości fali ≥ 700 nm i lampy rtęciowe, emitujące światło o długościach fali 546.1 nm i 435.8 nm.



390

Obserwator widzi przez wziernik obie ściany klina i ma za zadanie tak ustawić przysłony źródeł wzorcowych, aby jego zdaniem obie ściany klina były oświetlone tak samo. Współrzędne barwy odczytuje się z podziałek na przysłonach. Domeszanie światła wzorcowego do badanego umożliwia zmierzenie współrzędnych ujemnych.

Na podstawie dużej liczby eksperymentów z kolorymetrem zostały znalezione funkcje opisujące **współrzędne chromatyczne** — sygnały przekazywane przez czopki poszczególnych rodzajów otrzymamy, obliczając całki z iloczynów widma światła z tymi funkcjami.

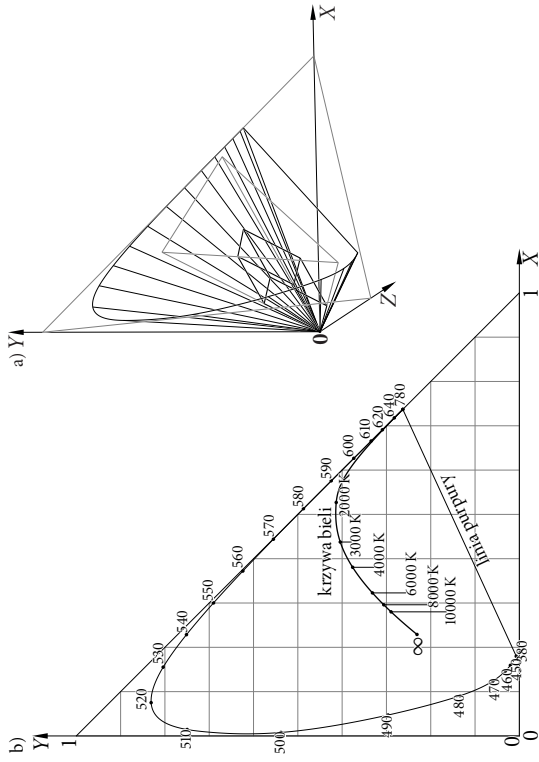


391

Diagram chromatyczności opracowany przez Międzynarodową Komisję Oświetleniową (CIE — *Commission Internationale de l'Éclairage*) w 1931 r. jest standardem, na którym opierają się używane w przemyśle układy współrzędnych w przestrzeni barw. Układ ten nazywa się CIE XYZ. „Światło” odpowiadające punktom, które tworzą układ odniesienia *nie istnieje*. Obszar barw widzialnych jest bryłą stożkową zawartą w dodatnim oktanccie przestrzeni \mathbb{R}^3 .

Przyjmując ustaloną moc światła, otrzymamy przekrój płaski tej bryły. Jest to obszar wypukły, którego brzeg składa się z krzywej tęczy (jej punkty reprezentują światło ściśle monochromatyczne) i linii purpury.

392



393

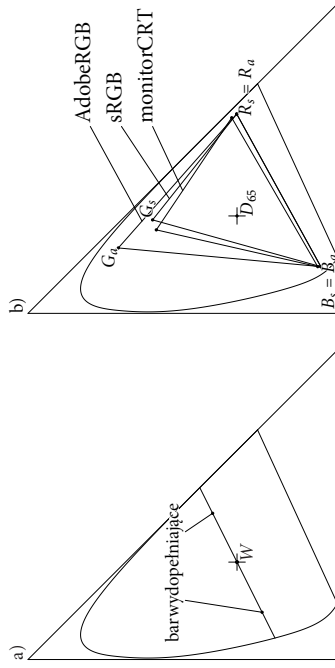
Krzywa bieli składa się z punktów reprezentujących barwy światła doskonale czarnego rozgrzanego do różnych temperatur. Za światło białe można przyjąć dowolny punkt tej krzywej, przy czym najczęściej jest to punkt D_{65} , odpowiadający temperaturze 6500 K, czyli temperaturze światła dziennego. Monitory mają możliwość wybierania temperatury, zwykle w zakresie od 6000 K do 7500 K.

Światło o dowolnej barwie można otrzymać przez zmieszanie światła białego ze światłem monochromatycznym albo z mieszaniną światła niebieskiego z czerwonym. W tym pierwszym przypadku możemy mówić o **dominującej długości fali**.

Nasylenie barwy reprezentowanej przez dowolny punkt jest ilorazem odległości tego punktu od przyjętego punktu bieli i długości odcinka przechodzącego przez ten punkt, którego jeden koniec jest punktem bieli, a drugi leży na brzegu obszaru barw widzialnych.

Barwy dopełniające leżą na odcinku przechodzącym przez punkt bieli po przeciwnych jego stronach i mają to samo nasycenie.

394



Barwy fizycznie realizowane na ekranie monitora powstają przez zmieszanie światła emitowanego przez elementy triad w każdym pikselu. Rysunek b) przedstawia trójkąt barw osiągalnych na typowym monitorze CRT oraz trójkąty barw reprezentowanych w układach współrzędnych określonych przez standardy **sRGB** (Microsoft & Hewlett-Packard, 1996) i **Adobe RGB** (Adobe Systems Inc, 1998).

395

Jeśli na otrzymanym obrazie występują barwy niemożliwe do odtworzenia na monitorze (reprezentowane przez punkty o ujemnych współrzędnych w układzie określonym przez wierzchołki trójkąta), to trzeba obraz przekształcić tak, aby zmiany były niedostrzegalne. Wzrost jest najmniej wyczułony na zmiany nasycenia barw — trzeba zatem dokonać **desaturacji** obrazu, przy czym korekcie trzeba poddać wszystkie barwy na obrazie.

396

Przebieg między układami współrzędnych CIE XYZ i sRGB składa się z trzech kroków; pierwszy jest przekształceniem liniowym, opisanym wzorem

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

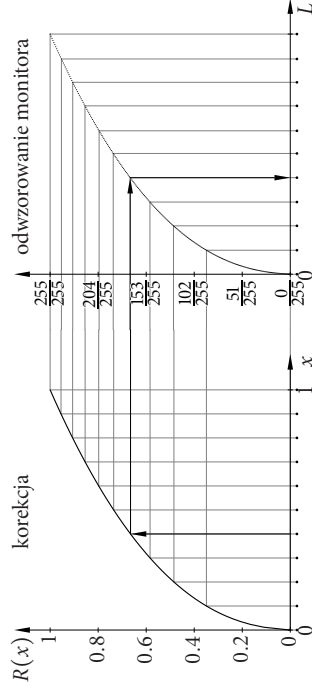
Drugi krok to obcięcie współrzędnych do przedziału $[0, 1]$. Trzeci krok to **korekcja gamma**. Moc światła emitowanego przez element triady w zależności od podanego na wejście sygnału jest opisana w przybliżeniu przez funkcję nieliniową $L(x) = cx^\gamma$, gdzie $\gamma \in [1.8, 2.8]$. Aby to skompensować, współrzędne (liniowo związane z mocą) r, g, b przekształca się za pomocą funkcji odwrotnej do L . W standardzie sRGB jest zamiast tego używana funkcja

$$R(x) = \begin{cases} 12.92x & \text{dla } x < 0.0031308, \\ 1.055x^{1/2.4} - 0.055 & \text{w przeciwnym razie.} \end{cases}$$

Otrzymane liczby z przedziału $[0, 1]$ można reprezentować przez osmiobitowe liczniki ułamków o mianowniku 255.

397

Dzięki wprowadzeniu opisanej nieliniowości odcienie barw ciemnych są reprezentowane dokładniej.



398

Przebieg między układami współrzędnych CIE XYZ i sRGB składa się z trzech kroków; pierwszy jest przekształceniem liniowym, opisanym wzorem

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

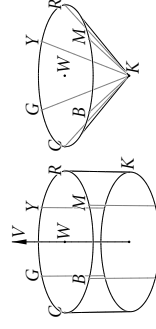
Drugi krok to obcięcie współrzędnych do przedziału $[0, 1]$. Trzeci krok to **korekcja gamma**. Moc światła emitowanego przez element triady w zależności od podanego na wejście sygnału jest opisana w przybliżeniu przez funkcję nieliniową $L(x) = cx^\gamma$, gdzie $\gamma \in [1.8, 2.8]$. Aby to skompensować, współrzędne (liniowo związane z mocą) r, g, b przekształca się za pomocą funkcji odwrotnej do L . W standardzie sRGB jest zamiast tego używana funkcja

$$R(x) = \begin{cases} 12.92x & \text{dla } x < 0.0031308, \\ 1.055x^{1/2.4} - 0.055 & \text{w przeciwnym razie.} \end{cases}$$

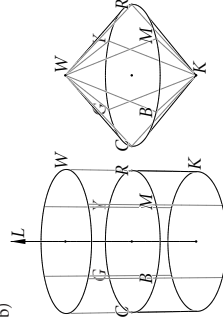
Otrzymane liczby z przedziału $[0, 1]$ można reprezentować przez osmiobitowe liczniki ułamków o mianowniku 255.

397

a)



b)



Odcień (*hue*) podaje się w stopniach. Nasylenie (*saturation*) przyjmuje wartości od 0 (na osi stożka, światło jest szare lub białe, odcień jest nieokreślony) do 1 (na bocznej powierzchni stożka). Współrzędna V (*value*) jest w zakresie od 0 (czarny) do 1 (główna podstawa stożka).

Układ HSL został wprowadzony w wyniku uwzględnienia faktu, że światło białe jest jaśniejsze niż najjaśniejsze światła o maksymalnym nasyceniu. Współrzędna L przyjmuje wartości z zakresu od 0 do 2.

399

400

Wizualizacja powierzchni zadanych niejawnie

Powierzchnia zadana niejawnie jest zbiorem miejsc zerowych funkcji (skalarnej) trzech zmiennych, tj. zbiorem rozwiązań równania

$$f(x, y, z) = 0,$$

lub ogólniej dowolną warstwicą takiej funkcji, określonej przez odpowiednio wybraną stałą c :

$$f(x, y, z) - c = 0.$$

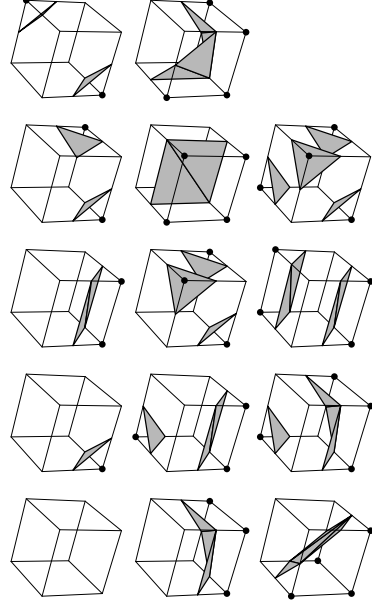
Można wykonywać obrazy takiej powierzchni za pomocą śledzenia promieni, ale można też znaleźć i narysować trójkąty przybliżające taką powierzchnię. Służy do tego wynaleziony w 1987 r. algorytm maszerujących sześciątów (ang. *marching cubes*).

401

Powierzchnię lub jej interesujący fragment należy obudować kostką (w ogólności równoległościenną), którą trzeba podzielić (równomiernie) na mniejsze kostki, tzw. **woksele**. Od ich rozmiarów zależy dokładność aproksymacji powierzchni przez trójkąty.

Kolejnym etapem jest stabilizowanie funkcji f w wierzchołkach wokseli, przez co powstaje trójwymiarowa tablica wartości. Jeśli funkcja jest dana jawnym wzorem lub pewnym algorytmem, to zadanie może wykonać (równoległe) GPU. Czasami tablica wartości funkcji jest dana — najczęściej w medycynie, gdzie funkcja f opisuje gęstość tkanek pacjenta (danymi otrzymanymi z tomografu) lub obraz czynnościowy (otrzymany za pomocą badania NMR lub przy użyciu gamma-kamery).

402



Dla danego wokselu szader np. geometrii może rozpoznać, który to przypadek, znaleźć wierzchołki trójkątów na krawędziach wokselu (przez liniową interpolację wartości funkcji $f - c$ na końcach krawędzi), skonstruować te trójkąty i je przekształcić, a potem wyprowadzić do etapu rasteryzacji.

404

Trójkąty są generowane dla tych wokseli, w których wierzchołkach wartości funkcji f przyjmują różne znaki, albo tych, w których wierzchołkach są wartości funkcji większe i mniejsze od liczby c .

Przy założeniu, że 0 traktujemy jak liczby dodatnie (albo ujemne), jest $2^8 = 256$ możliwych rozłożeń tych znaków. Ale, jest tylko 14 istotnie różnych rozkładów znaków w wierzchołkach wokselu, każdy układ znaków przez zanegowanie, obrót lub odbicie symetryczne można sprowadzić do jednego układu z tych 14.

Można stworzyć tablicę (o długości 128 — jeśli najbardziej znaczący bit układu jest jedynką, to negujemy wszystkie bity), której elementy są macierzami odpowiednich przekształceń.

403

Kłopot z tym algorytmem polega na tym, że nie zawsze trójkąty w wkselach mających wspólną ścianę będą miały wspólną krawędź. „Nieszczelne” sklejenie trójkątów może się zdarzyć, jeśli na tej ścianie funkcja $f - c$ będzie miała wartości o przeciwnych znakach na każdej krawędzi (czyli będzie mieć ten sam znak na końcach każdej przekątnej ściany).

W zasadzie można by ten problem rozwiązać, badając znak w środku ściany, ale to nie jest możliwe, jeśli jest tylko dana tablica wartości funkcji f .

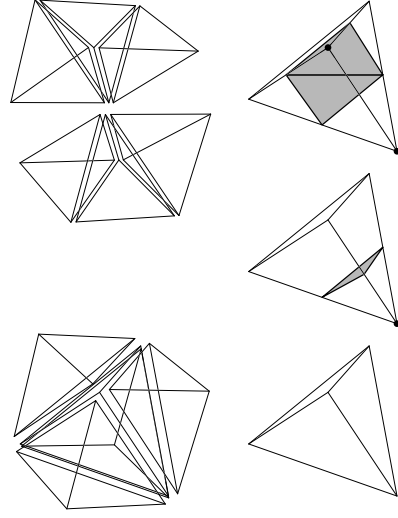
Pewnym rozwiązaniem, przy okazji znacznie upraszczającym algorytm (ale prowadzącym do zwiększenia liczby generowanych trójkątów) jest zastąpienie „maszerujących sześciątów” „maszerującymi czworoscianami”.

405

Wksel (tj. równoległoscian) można podzielić na 5 czworoscianów (w tym jeden foremny), przy czym istnieją dwa sposoby takiego podziału i wkselse mające wspólną ścianę muszą być podzielone różnymi sposobami, aby ściany czworoscianów były wspólne.

Dlatego lepszym pomysłem jest podzielenie wksela trzema płaszczyznami na 6 czworoscianów. Zrobienie tego trzema rodzinami płaszczyzn równoległych wytwarza czworosciany, które mają całe ściany wspólne.

406



W każdym czworoscianie trzeba wygenerować tylko 0, 1 albo 2 trójkąty, przy czym ich boki będą się pokrywać z bokami trójkątów w czworoscianach sąsiednich.

407

Obraz powierzchni, czyli trójkąty, powinien być oświetlony, a do tego potrzebny jest wektor normalny w każdym punkcie. Oczywiście, dla każdego trójkąta można wprowadzić wektor normalny jego płaszczyzny, ale wtedy na obrazie będzie widać, że to trójkąty, a nie powierzchnia gładka.

Jeśli funkcja f jest dana jawnym wzorem, to można wprowadzić wzory opisujące jej pochodne cząstkowe i razem z każdym wierzchołkiem wprowadzić unormowany gradient funkcji f .

Dla funkcji f danej w postaci tablicy wypada dla każdego wierzchołka zajrzeć do wkseli sąsiednich: na podstawie wartości funkcji f w wierzchołkach wksela danego i sąsiednich trzeba skonstruować wielomian interpolacyjny (np. stopnia 1 ze względu na każdą zmienną) i obliczyć jego gradient.

Zagładanie do sąsiadów jest konieczne, aby w każdym wspólnym wierzchołku trójkątów wprowadzony wektor „normalny” był określony jednoznacznie.

408

Swobodna deformacja

Platy Béziera są określone przy użyciu iloczynu tensorowego par wielomianów: $(B_i^n \otimes B_j^m)(u, v) = B_i^n(u)B_j^m(v)$. Za pomocą iloczynu tensorowego trójek wielomianów można określić przekształcenie obszaru trójwymiarowego (np. kostki $[0, 1]^3$) zwane **swobodną deformacją** (ang. *free-form deformation, FFD*):

$$f(r, s, t) = \sum_{i=0}^n \sum_{j=0}^m \sum_{k=0}^l f_{ijk} B_i^n(r) B_j^m(s) B_k^l(t),$$

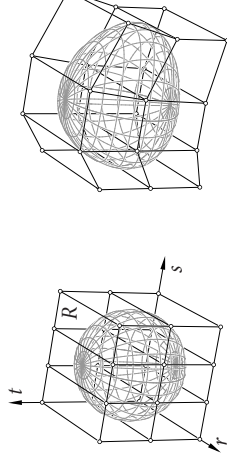
Punkty kontrolne $f_{ijk} \in \mathbb{R}^3$ tworzą siatkę trójwymiarową. Niech R oznacza dowolny równoległoscian. Można wprowadzić lokalny układ współrzędnych r, s, t , w którym ten równoległoscian jest kostką jednostkową (zatem jeden z wierzchołków jest początkiem tego układu, a wektory r, s, t wychodzących z niego krawędzi są wersorami osi). Mając dowolny punkt o współrzędnych (x, y, z) w układzie świata, możemy obliczyć wektor (r, s, t) , a następnie znaleźć punkt $f(r, s, t)$ odkształconego obiektu.

409

Jeśli punkty kontrolne są określone wzorem

$$f_{ijk} = f_{000} + \frac{i}{n}r + \frac{j}{m}s + \frac{k}{l}t,$$

w którym punkt f_{000} jest początkiem układu r, s, t , to funkcja f jest przekształceniem afinicznym — przejściem od układu współrzędnych (r, s, t) do układu świata (x, y, z) . Zatem złożenie dwóch przekształceń — przejścia od układu (x, y, z) do układu (r, s, t) z funkcją f jest przekształceniem tożsamościowym.



410

Własności FFD:

Własność otoczki wypukłej: Obrazem równoległoscianu R jest tzw. **bryła Béziera** (*Bézier volume*) zawarta w otoczce wypukłej zbioru punktów f_{ijk} .

Własności interpolacyjne: Brzeg bryły Béziera (w zasadzie) składa się z sześciu płatów Béziera.

Algorytm de Casteljau, a także wszystkie inne algorytmy przetwarzania krzywych Béziera (schemat Hornera, podwyższanie stopnia) można stosować do przetwarzania swobodnej deformacji, analogicznie jak dla płatów Béziera.

411

Przekształcenie afiniczne punktów kontrolnych swobodnej deformacji jest równoznaczne ze złożeniem deformacji z tym przekształceniem (a zatem możemy przesuwać, obracać i skalować zdeformowane obiekty, przekształcając odpowiednio siatkę deformacji).

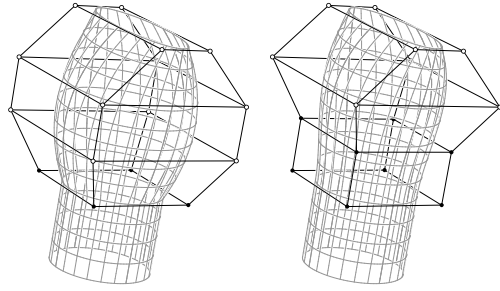
Pochodne przekształcenia f możemy obliczyć na podstawie wektorów

$$\Delta_1 f_{ijk} = f_{i+1,j,k} - f_{ijk}, \Delta_2 f_{ijk} = f_{i,j+1,k} - f_{ijk}, \Delta_3 f_{ijk} = f_{i,j,k+1} - f_{ijk},$$

za pomocą wzorów podobnych do tych, które opisują pochodne cząstkowe płatów Béziera.

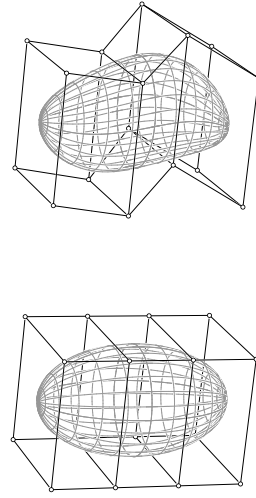
Gładkie łączenie deformacji może się przydać, gdy ma być zdeformowana tylko pewna część obiektu (znajdująca się wewnątrz równoległoscianu R) lub gdy różne części mają być poddane różnym odkształceniom.

412

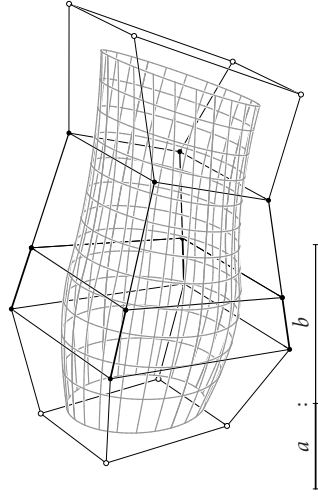


413

Objętości odkształconych brył będą zachowane, jeśli siatkę początkową (która dzieli równoległościان R na jednakowe równoległościان) zmienimy, wybierając *jedną* z trzech indeksów, np. k , i dla przesuwając każdego k wszystkie punkty f_{ijk} w płaszczyźnie, w której leżą. To wynika ze znanej już Archimedesowi zasady Cavalieriego (1635 r.).



415



414

Pierwszy pomysł swobodnej deformacji, przedstawiony wyżej, opublikowali w 1996 r. Sederberg i Parry. Oczywiście, zamiast wielomianów, można użyć tensorowych iloczynów funkcji sklepanych (np. B-sklepanych).

Nową jakość wprowadziła Sabine Coquillart w 1990 r., opracując **rozszerzoną swobodną deformację** (*extended free-form deformation, EFFD*). Przekształcenie obszaru trójwymiarowego jest złożeniem dwóch przekształceń, z których drugie jest swobodną deformacją, a pierwsze jest odwrotnością (w ogólności innej) swobodnej deformacji. Mamy więc wzór

$$f(x, y, z) = g(h^{-1}(x, y, z)),$$

w którym g i h są swobodnymi deformacjami. Szczególnym przypadkiem EFFD jest zwykła FFD, w której h jest przekształceniem afinicznym.

416

Aby określić EFFD, należy rozmieścić w przestrzeni wierzchołki dwóch trójwymiarowych siatek (w ogólności nie muszą one określać przekształceń tego samego stopnia), aby określić dwa przekształcenia kostki $[0, 1]^3$ (lub innego prostopadłościanu) w przestrzeń. Aby przekształcenie było dobrze określone, przekształcenie h *nie musi* być różnowartościowe; wystarczy, aby punkty „sklejone” przez przekształcenie h były również sklejone przez przekształcenie g .

Algorytm EFFD składa się z następujących kroków: odkształcany obiekt (np. powierzchni) należy odpowiednio rozdzielić (np. podzielić na dostatecznie małe trójkąty). Otrzymane punkty P_i (wierzchołki trójkątów) trzeba poddać przekształceniu h^{-1} , czyli numerycznie rozwiązać równania $h(x_i) = P_i$.

W przypadku, gdy rozwiązanie nie jest jednoznaczne należy wybrać jedno rozwiązanie. Ten etap nazywa się **zamrażaniem** obiektu w bryle przekształcenia h .

Końcowy etap obliczenia polega na przekształceniu punktów x_i przy użyciu funkcji g .