Modelowanie bryłowe

W ogólności zakładamy, że jesteśmy w stanie określić dowolny punkt bryły, tzn. nie tylko brzegowy, ale również wewnętrzny. Większość metod nie pozwala jednak na to ostatnie. Ponadto wyraźnie rozgraniczamy modelowanie bryłowe same w sobie, tzn. jednorodne modele pojedynczych brył określonej klasy, od metod złożonych , tzn. używających dodatkowo zbioru operacji (CSG) lub reprezentacji grafowych do określenia połączeń pomiędzy składowymi (b-rep, podziały przestrzenne)

1 Modele jednorodne

Można je podzielić na kilka podklas [**?**]

(i) bryły parametryczne, w szczególności trójkubiczne bryły Hermite'a

(ii) instancje

(iii) reprezentacje z przesuwaniem

(iv) bryły definiowane przez odwzorowania nieliniowe

Ze względu na szczupłość czasu przeznaczonego na te zagadnienia omówimy tylko bryly parametryczne i przykład dla reprezentacji z przesuwaniem.

1.1 Bryły parametryczne

W sposób naturalny rozszerzają one pojęcie powierzchni parametrycznej, tzn. są dane przez układ równań

$$b(u,v,w) \equiv \begin{cases} x &= x(u,v,w) \\ y &= y(u,v,w) \\ z &= z(u,v,w) \end{cases} \qquad u,v,w \in [0,1] \tag{1.1}$$

Dla ograniczonego zbioru parametrów do kostki $[0,1]^3$ otrzymujemy, tzw. *hiperpłat bryły*. Brzeg tak określonego hiperpłata jest określony przez 8 punktów narożnikowych, 12 krzywych krawędziowych oraz 6 płatów brzegowych. Podobnie jak w przypadku powierzchni warunki brzegowe można też wyspecyfikować inaczej, np. przez podanie wektorów stycznych i wektorów skrętu. Notacja jest analogiczna jak dla powierzchni, tzn. p_{000} oznacza narożnik $p(0,0,0)$, p_{u00} krzywą $p(u,0,0)$, a p_{uv0} płat $p(u,v,0)$, np. bryła prostokątna jest dana układem

$$b(u,v,w) \equiv \begin{cases} x &= (b-a)u + a \\ y &= (d-c)v + c \\ z &= (f-e)w + e \end{cases} \qquad u,v,w \in [0,1]$$

i jest prostopadłościanem o bokach równoległych do płaszczyzn układu i zaczepionym pomiędzy punktami (a,c,e), (b,d,f).

Bryły trójkubiczne Hermite'a Określa je iloczyn tensorowy

$$p(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}a_{ijk}u^iv^jw^k \qquad u,v,w\in[0,1]\;\; a_{ijk}\in R^3 \tag{1.2}$$

co daje w sumie 192 współczynników. Postać geometryczna podobnie jak w przypadku powierzchni korzysta z trójkowych iloczynów baz Hermite,a, które tworzą nowy układ bazowy, tzn. przyjmuje ona formę:

$$p(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}b_{ijk}b_i(u)b_j(v)b_k(w)$$

Każdy z ośmiu punktów reprezentujących końce krzywych krawędziowych jest indeksowany zgodnie z kierunkami wzrostu parametrów, tzn. od p_{000} do p_{111}. Punktowi temu przypisany jest wektor położenia, trzy wektory stycznych do trzech krawędzi wychodzących z niego, trzy wektory skrętu w kierunkach tych krawędzi oraz jeden dodatkowy wektor p^{uvw} określający szybkość zmian wektora skrętu p^{uv}. Daje to osiem wektorów w każdym z ośmiu punktów. Ponieważ w równaniu powyżej mamy 64 współczynniki b_{ijk} więc można je powiązać z 64 wektorami przypisanymi końcom krawędzi. Ustalmy cztery macierze B_k, $k=1,\ldots,4$ odpowiadające ustalonemu indeksowi dla parametru w, tzn. $B_k=[b_{ijk}]_{i=1,\ldots,4}^{j=1,\ldots,4}$. Rozlokujmy w nich 64 wektory tak, że B_1 i B_2 są odpowiednio macierzami geometrii dla powierzchni stałych parametrów $w=0$ i $w=1$, a dwie pozostałe są pochodnymi tych powierzchni ze względu na parametr w. Mamy więc

$$B_1=\begin{bmatrix} p_{000} & p_{100} & p^u_{000} & p^u_{100}\\ p_{010} & p_{110} & p^u_{010} & p^u_{110}\\ p^v_{000} & p^v_{100} & p^{uv}_{000} & p^{uv}_{100}\\ p^v_{010} & p^v_{110} & p^{uv}_{010} & p^{uv}_{110}\end{bmatrix} \qquad B_2=\begin{bmatrix} p_{001} & p_{101} & p^u_{001} & p^u_{101}\\ p_{011} & p_{111} & p^u_{011} & p^u_{111}\\ p^v_{001} & p^v_{101} & p^{uv}_{001} & p^{uv}_{101}\\ p^v_{011} & p^v_{111} & p^{uv}_{011} & p^{uv}_{111}\end{bmatrix}$$

$$B_3=\begin{bmatrix} p^w_{000} & p^w_{100} & p^{uw}_{000} & p^{uw}_{100}\\ p^w_{010} & p^w_{110} & p^{uw}_{010} & p^{uw}_{110}\\ p^{vw}_{000} & p^{vw}_{100} & p^{uvw}_{000} & p^{uvw}_{100}\\ p^{vw}_{010} & p^{vw}_{110} & p^{uvw}_{010} & p^{uvw}_{110}\end{bmatrix} \qquad B_4=\begin{bmatrix} p^w_{001} & p^w_{101} & p^{uw}_{001} & p^{uw}_{101}\\ p^w_{011} & p^w_{111} & p^{uw}_{011} & p^{uw}_{111}\\ p^{vw}_{001} & p^{vw}_{101} & p^{uvw}_{001} & p^{uvw}_{101}\\ p^{vw}_{011} & p^{vw}_{111} & p^{uvw}_{011} & p^{uvw}_{111}\end{bmatrix}$$

Wtedy możemy zapisać

$$p(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}b_{ijk}b_i(u)b_j(v)b_k(w)$$

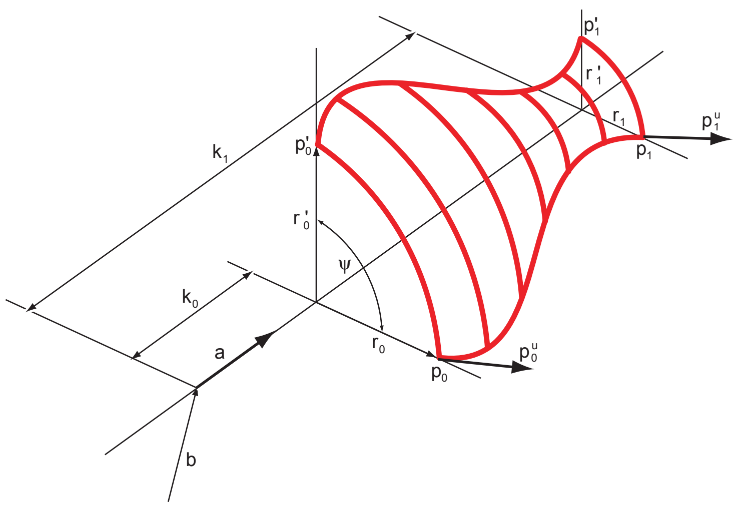
$$=\sum_{i=0}^{3}\sum_{j=0}^{3}b_i(u)b_j(v)[b_{ij1}b_1(w)+b_{ij2}b_2(w)+b_{ij3}b_3(w)+b_{ij4}b_4(w)]$$

$$=\sum_{k=0}^{3}b_k(w)\left[\sum_{i=0}^{3}\sum_{j=0}^{3}(b_{ijk}b_i(u)b_j(v)\right]$$

$$= W^T \cdot M_H^T \cdot \begin{bmatrix} V^T \cdot M_H^T \cdot B_1 \cdot M_H \cdot U \\ V^T \cdot M_H^T \cdot B_2 \cdot M_H \cdot U \\ V^T \cdot M_H^T \cdot B_3 \cdot M_H \cdot U \\ V^T \cdot M_H^T \cdot B_4 \cdot M_H \cdot U \end{bmatrix}$$

1.2 Powierzchnie obrotowe jako przykład reprezentacji z przesuwaniem

Powierzchnie te są generowane przez krzywą płaską zwaną *linią profilu*, która zaczepiona w danym punkcie obraca się względem ustalonej *osi obrotu*. Za [**?**] rozpatrzmy ogólną sytuację, w której dana jest krzywa Hermite,a $[p_0 \ p_1 \ p_0^t \ p_1^t]$, wektor jednostkowy a oznaczający kierunek obrotu, punkt b, przez który oś obrotu przechodzi, oraz ψ oznaczający kąt obrotu (rys. 1.1) Z rysunku dostajemy, że $a \cdot r_0 = a \cdot r_1 = 0$ oraz $b + k_0 a + r_0 = p_0$ i $b + k_1 a + r_1 = p_1$.



Rysunek 1.1: Konstrukcja powierzchni obrotowej.

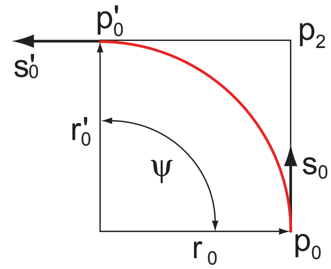
Mnożąc dwa ostatnie równania lewostronnie przez a otrzymujemy $k_0 = a \cdot (p_0 - b)$ oraz $k_1 = a \cdot (p_1 - b)$. Z rysunku dostajemy $p_0^{'} = b + k_0 a + r_0^{'}$, $p_1^{'} = b + k_1 a + r_1^{'}$. Zatem musimy znaleźć $r_0^{'}$ i $r_1^{'}$. Również z rysunku wynika , że kierunek stycznej do powierzchni obrotu w punkcie p_0 wyraża się wzorem $s_0 = \frac{r_0 \times a}{|r_0 \times a|}$ (układ lewoskrętny). Obliczmy współrzędne punktu $r_0^{'}$ w układzie wersorów $(\frac{r_0}{|r_0|}, s_0)$. Wtedy z własności obrotu dostajemy, że

$$r_0^{'} = |r_0| \frac{r_0}{|r_0|} cos\psi + |r_0| s_0 sin\psi = r_0 cos\psi + |r_0| \frac{r_0 \times a}{|r_0 \times a|} sin\psi$$

Analogicznie znajdujemy

$$r_1^{'} = |r_1| \frac{r_1}{|r_1|} cos\psi + |r_1| s_0 sin\psi = r_1 cos\psi + |r_1| \frac{r_1 \times a}{|r_1 \times a|} sin\psi$$

Ponadto $s_0^{'} = s_0 cos\psi - \frac{r_0}{|r_0|} sin\psi$ (rys. 1.2) oraz $s_1 = s_0$ i $s_1^{'} = s_0^{'}$. Trzeba znaleźć jeszcze długości wektorów stycznych do wycinka obrotu, tzn krzywej wyznaczonej przz punkty p_0 i $p_0^{'}$ (s_0 i $s_0^{'}$ są wektorami jednostkowymi i wyznaczają tylko kierunki stycznych). Przyjmijmy ,że odpowiada za to parametr w, a krzywa oznaczająca linię profilu związana jest z parametrem v. Z rysunku (1.2) oraz wzorów dla powierzchni stożkowych w postaci Hermite'a (zob. ćwiczenia po wykładzie piątym) wynika, że $p_2 = p_0 + \frac{|p_0^{'} - p_0|}{|s_0 + s_0^{'}|} s_0$ oraz



Rysunek 1.2: Wektory styczne do powierzchni obrotowej.

$$p_{00}^w = 4|r_0| \left[\frac{1 - cos(\frac{\psi}{2})}{sin(\frac{\psi}{2})} \right] \left[\frac{p_2 - p_0}{|p_2 - p_0|} \right]$$

$$p_{01}^w = 4|r_0| \left[\frac{1 - cos(\frac{\psi}{2})}{sin(\frac{\psi}{2})} \right] \left[\frac{p_2^{'} - p_0}{|p_2^{'} - p_0|} \right]$$

(p_{00} odpowiada punktowi p_0, p_{01} punktowi $p_0^{'}$) Pozostaje określić wektor p_{01}^u, który jest wynikiem obrotu wektora p_0^u (oczywiście ich długości są takie same). Zróbmy to w układzie wersorów $(a, \frac{r_0^{'}}{|r_0^{'}|})$. Jasne jest, że $p_{01}^u \cdot a = p_0^u \cdot a$. Podobnie rzut wektora p_{01}^u na kierunek $r_0^{'}$ jest równy rzutowi wektora p_0^u na kierunek r_0. Daje to w sumie

$$p_{01}^u = (p_0^u \cdot a)a + \left(p_0^u \cdot \frac{r_0}{|r_0|} \right) \left(\frac{r_0^{'}}{|r_0^{'}|} \right)$$

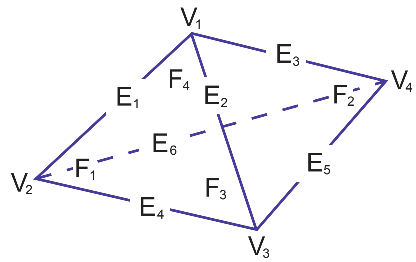
Analogicznie znajdujemy wektor p_{11}^u. Wektory skrętu przyjmujemy jako zerowe.

2 Modelowanie złożone

2.1 Reprezentacje brzegowe

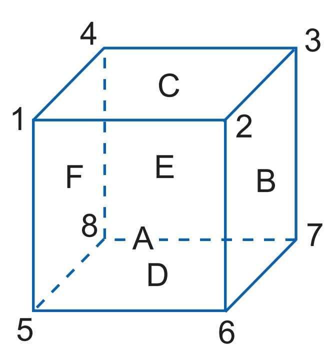
2.1.1 Wstęp

Rozważamy, bardzo ogólnie (nie uwzględniając np. orientacji), reprezentacje figur wielościennych. Modelowanie tego typu polega na określeniu połączeń pomiędzy ścianami,



Rysunek 2.1: Wierzchołki, krawędzie i ściany czworościanu.

krawędziami i wierzchołkami obiektu. Rozpatrzmy czworościan jak na rysunku 2.1 Sposób jego jednoznacznego określenia może być różny. Możemy np. jako bazę wybrać wierzchołki, następnie do każdego z nich przypisać sąsiadujące z nim bezpośrednio inne wierzchołki, krawędzie stykające się z nim oraz ściany przylegające do tych krawędzi. Dla V_1 wyglądać to może np. tak:
$V_1\{V_2, V_3, V_4, E_1, E_2, E_3, F_1, F_2, F_4\}$
Możemy jednak postąpić inaczej, mianowicie rozpocząć od krawędzi, następnie przypisać każdej z nich dwa wierzchołki, które ją tworzą, cztery krawędzie stykające się z nią w tych wierzchołkach oraz dwie ściany do niej przylegające, np.
$E_1\{V_1, V_2, E_2, E_3, E_4, E_6, F_1, F_4\}$
Trzecim możliwym podejściem jest wyjście od ściany, następnie przypisanie jej wierzchołków, którę ją określają, krawędzi ograniczjących ją i ścian z nią sąsiadujących, np.
$F_1\{V_1, V_2, V_3, E_1, E_4, E_2, F_2, F_3, F_4\}$
Widać, że w wielu przypadkach dane są obarczone redundancją informacji.Nieco lepszym podejściem jest określenie wierzchołków w jednej macierzy oraz ich połączeń w znanej z teorii grafów macierzy przyległości (w tym przypadku dla wierzchołków). Dla sześcianu z rysunku 2.2 taka macierz ma postać (numery indeksów odpowiadają numerom wierzchoł-



Rysunek 2.2: Wierzchołki sześcianu pozwalają całkowicie określić go przestrzennie konstruując macierz przyległości (poniżej).

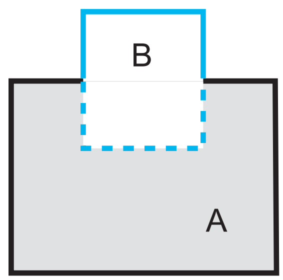
ków):

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Analogiczne macierze można budować dla krawędzi i ścian.

2.1.2 CSG

Boolowske operatory regularne Ze względu na fakt, iż operujemy zwykle na figurach zawierających wszystkie swoje punkty brzegowe to wymagamy aby operacje boolowskie na takich figurach dawały wynik również posiadający tę własność. Niestety zwykłe operacje sumy, przekroju i różnicy nie zawsze spełniają to założenie. Rysunek 2.3 pokazuje prosty



Rysunek 2.3: Nie wszystkie punkty brzegowe należą do różnicy prostokątów $A \setminus B$

przykład takiej sytuacji. W związku z tym wprowadzamy zbiór operacji regularyzujących zwykłe operatory $\cup$, $\cap$, $\setminus$. Wprowadźmy następujące definicje

1. $\mathrm{R}(A) := \overline{Int(A)}$
2. Zbiór A jest *regularny* gdy $A = \mathrm{R}(A)$
3. Niech *op* oznacza jedną z operacji $\cup$, $\cap$, $\setminus$. Wtedy $A\, op^*\, B := \overline{Int(A\, op\, B)}$

Fakt 2.1 *$R(A)$ jest jednorodny wymiarowo dla każdego zbioru A*

Posługując się zbiorem zwykłych operacji boolowskich na brzegach i wnętrzach dwóch figur możemy operatory $\cup^*$, $\cap^*$, $\setminus^*$ określić jednoznacznie przez sumę pewnego podzbioru tych operacji, tzn.

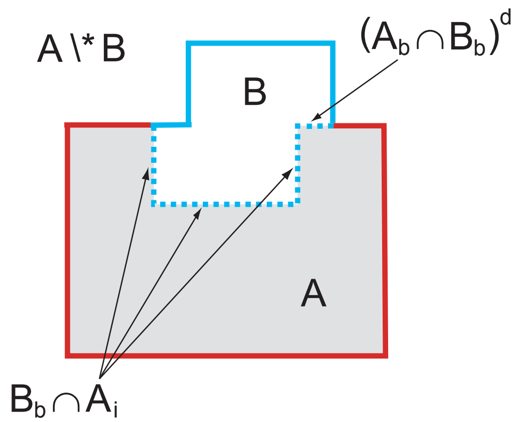
$$A \cup^* B = \bigcup\{A_i \cap B_i,\ A_i \setminus B,\ B_i \setminus A,\ A_b \setminus B,\ B_b \setminus A,\ (A_b \cap B_b)^s\}$$

gdzie $(A_b \cap B_b)^s$ oznacza fragment wspólnego brzegu dla którego A_i i B_i znajdują się po tej samej jego stronie.

$$A \cap^* B = \bigcup \{A_i \cap B_i,\ A_b \cap B_i,\ B_b \cap A_i,\ A_b \setminus B,\ (A_b \cap B_b)^s\}$$

$$A \setminus^* B = \bigcup \{A_i \setminus B,\ B_b \cap A_i,\ A_b \setminus B,\ (A_b \cap B_b)^d\}$$

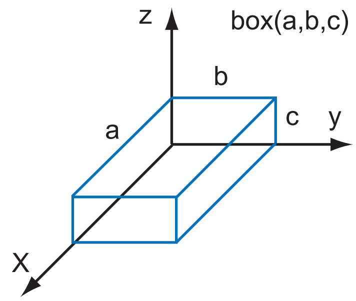
gdzie $(A_b \cap B_b)^d$ oznacza fragment wspólnego brzegu dla którego A_i i B_i znajdują się po jego przeciwnych stronach. Przypadek różnicy regularnej zachowuje się źle w sytuacji gdy intresują nas normalne do brzegu figury (2.4). W takiej sytuacji na sumie obszarów



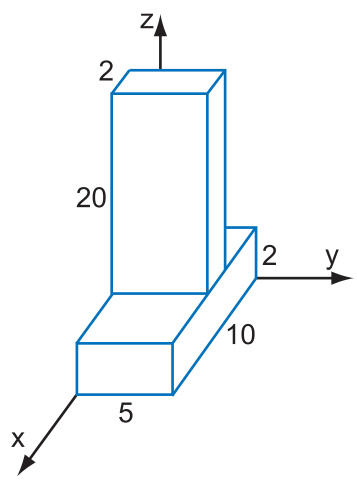
Rysunek 2.4: Normalne na fragmencie wykropkowanym wymagaja zmiany po operacji różnicy regularnej

$(A_b \cap B_b)^d$ oraz $B_b \cap A_i$ trzeba po wykonaniu operacji zmienić orientację brzegu (tzn. zwrot normalnej).

Obiekty CSG Obiekt CSG jest zbudowany ze zbioru standardowych prymitywów w połączeniu z użyciem skończonej rodziny regularnych operacji boolowskich oraz zbioru standardowych operacji geometrycznych tzn. zwykle translacji, obrotu i skali. Standardowe prymitywy to zwykle prostopadłościan, sfera, walec, stożek i torus. Ponieważ obiekt CSG ma powstać jako wynik operacji na skończonym zbiorze prymitywów, więc oczywiście musimy zdefiniować modele dla każdego z prymitywów standardowych. Rysunek 2.5 pokazuje przykład modelu prostopadłościanu. Rozpatrzmy figurę złożoną z dwóch prostopadłościanów dających w wyniku T-kształtną bryłę (rys. 2.6). wtedy posługując się instancją modelu $box(a, b, c)$ oraz operacjami translacji wzdłuż osi układu współrzednych możemy ten obiekt opisać jako
box(10,5,2) $\cup^*$ **z_trans(x_trans(box(2,5,20),4),2)**
Ze względu na "zstępujący" charakter dołączania kolejnych operacji do wyniku poprzedniej wygodną konwencją dla opisu tego procesu jest drzewo binarne, którego liście są instancjami obiektów, a pozostałe węzły reprezentują operacje. Drzewo dla rozpatrywanego



Rysunek 2.5: Model prostopadłościanu.

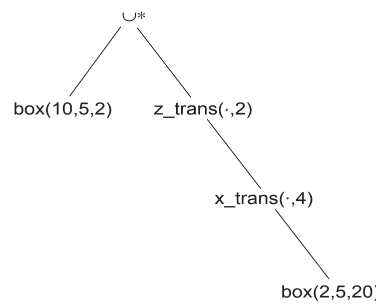


Rysunek 2.6: Bryła CSG będąca sumą dwóch przesuniętych instancji prostopadłościanu.

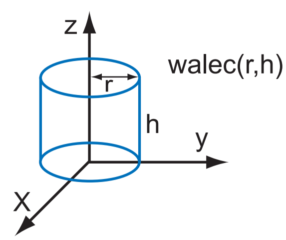
przykładu przedstawia rysunek (2.7). Określmy teraz model walca (rys. 2.8). i posługując się jego instancją wytnijmy dziurę w wyższym prostopadłościanie (rys. 2.9). Aby to zrobić musimy od otrzymanej wcześniej sumy odjąć obróconą i przesuniętą instancję walca. Ciąg operacji będzie następujący:

1. walec(1,2)
2. obrót wokół osi y o kąt 90^o, tzn. y_rot(·,90)
3. przesunięcie wzdłuż osi x o 4 jednostki,tzn. x_trans(·,4)
4. przesunięcie wzdłuż osi y o 2.5 jednostki,tzn. y_trans(·,2.5)
5. przesunięcie wzdłuż osi z o 12 jednostek,tzn. z_trans(·,12)

co daje odjemnik postaci
z_trans(y_trans(x_trans(y_rot(walec(1,2),90),4),2.5),12).
Drzewo dla nowego obiektu CSG przedstawione jest na rysunku 2.10.



Rysunek 2.7: Drzewo binarne odpowiadające bryle z rysunku 2.6.

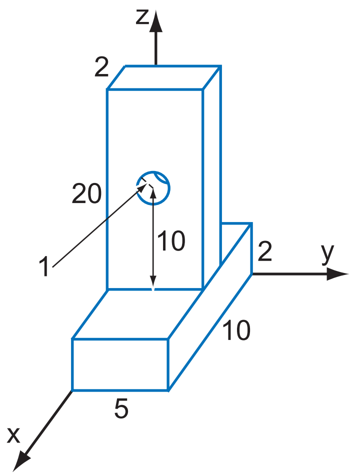


Rysunek 2.8: Model walca.

Klasyfikacja punktu względem obiektu CSG Jest to jedno z klasycznych zagadnień obok analogicznych dotyczących przecinania obiektu przez proste (w ogólności rozpatruje się krzywe), przecinania obiektu przez powierzchnię czy też wreszcie przecinania się dwóch obiektów CSG. W rozpatrywanym przypadku dany jest punkt w przestrzeni oraz ustalony obiekt CSG, tzn. drzewo, które go opisuje. Pytamy czy punkt należy do obiektu (jeśli tak to czy należy do jego wnętrza czy do brzegu), czy też leży poza nim. Pierwsze możliwe rozwiązanie jest następujące: Przesuwamy punkt w dół drzewa począwszy od $root$ zgodnie z regułami:

1. jeśli węzeł jest operacją boolowską to prześlij punkt bez zmainy do dwóch jego potomków
2. jeśli węzeł jest operacją geometryczną to zastosuj do punktu operację odwrotną do danej i tak przekształcony prześlij do dwóch potomków węzła
3. jeśli węzeł jest liściem (tzn. instancją obiektu) to skalsyfikuj punkt otrzymując w wyniku on, off lub in.

Następuje teraz druga faza - cofanie się po węzłach do $roota$. Wtedy jeśli jedna z wartości ze zbioru $\{in, on, off\}$ jest przesłana do węzła nadrzędnego będącego operacją geome-

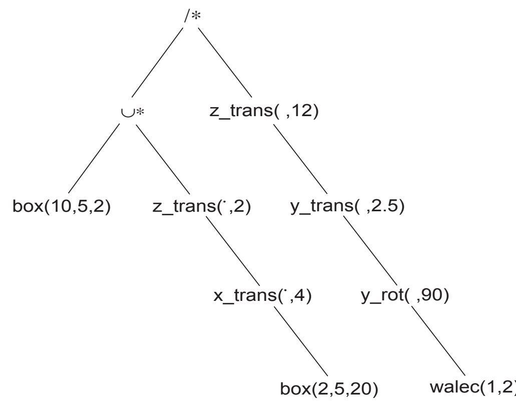


Rysunek 2.9: Bryła CSG będąca różnicą bryły z rysunku 2.6 oraz przesuniętej i obróconej instancji walca.

tryczną to przesyłamy ją wyżej bez zmiany. W przypadku dojścia do węzła zawierającego operator boolowski musimy wykonać odpowiednio zdefiniowane sumowanie na wartościach z dwóch potomków węzła. Reguły takiego sumowania są łatwe do odgadnięcia (np. dla sumy regularnej $in + in = in$, $on + in = in$ itd.). Czytelnik może w ramach ćwiczenia sprządzić pełne tabele takiego sumowania dla wszystkich operacji regularnych. Okaże się wtedy, że kłopot sprawia operacja $on + on$. Dlaczego tak jest pokazuje próba przyporządkowania punktu $(5, 2.5, 2)$ z powyższego przykładu. Inaczej niż w większości sytuacji dających $on + on = on$ w tym przypadku mamy $on + on = in$. Zatem poprawne zweryfikowanie wyniku takiej operacji musi pociągać za sobą skorzystanie z jakiś dodatkowych kryteriów. Możliwym rozwiązaniem jest zbadanie struktury sąsiedztwa badanego punktu, przy czym przez *sąsiedztwo* rozumiemy przekrój możliwie małego otoczenia punktu z obiektem. Przykładowo, jeżeli punkt należy do wnętrza ściany obiektu to sąsiedztwo jest półkulą zawartą w obiekcie o brzegu zawartym w danej ścianie. W sytuacji gdy punkt należy do wnętrza krawędzi to dostaniemy sąsiedztwo w kształcie przekroju klina i kuli, itd. Analiza wszystkich możliwych przypadków zależy silnie od dopuszczalnej topologii brył (zob. [2]). Mając już określone sąsiedztwa będziemy przeprowadzć operację na sąsiedztwach zamiast na wartościach typu in, on, off. Dla zarysowania rozwiązania oznaczmy przez S_L i S_R potomków danego węzła, a przez S sąsiedztwo wynikowe. Wtedy przykładowo, dla operacji sumowania możemy stwierdzić, że

1. jeżeli S_L jest pełną kulą to $S = S_L$, jeśli S_L jest puste to $S = S_R$

2. jeżeli oba sąsiedztwa są typu półkuli to wynik będzie sąsiedztwem typu klina, chyba że ściany mają część wspólną będącą ich fragmentem (przypadek problematycznego punktu z powyższego przykładu). Wtedy zależnie od orientacji ścian S będzie typu półkuli lub będzie pełną kulą



Rysunek 2.10: Drzewo binarne odpowiadające bryle z rysunku 2.9

Widać na pierwszy rzut oka, że efektywna implementacja sąsiedztw i operacji na nich stanowi niełatwe wyzwanie. Jeszcze trudniejsze są oczywiście przypadki innych klasyfikacji [2].

2.1.3 b-rep

Idea polega na zbudowaniu reprezentacji bryły jako zorganizowanej w jakiś sposób rodziny podzbiorów powierzchni, np. dla brył wielościennych potrzebujemy informacji o równaniach płaszczyzn, do których należą ściany oraz o punktach leżących na tych płaszczyznach określających uporządkowany zbiór wierzchołków każdej ze ścian. Oznaczmy przez $K^{m,n}$, $m \leqslant n$, m-wymiarowy, ograniczony, domknięty podzbiór przestrzeni R^n. Wtedy możemy go rozbić na punkty należące do brzegu oraz należące do wnętrza. Wyrażamy to jako

$$R^{m,n} = [B^{m-1,n}, I^{m,n}]$$

gdzie $B^{m-1,n}$ oznacza $m-1$-wymiarowy brzeg, a $I^{m,n}$ m-wymiarowe wnętrze. Każdy punkt $P \in R^3$ jednoznacznie daje się przypisać do któregoś z tych obszarów lub do $R^3 \setminus K^{m,n}$. Ponadto dla $m = n$ wnętrze jest jednoznacznie określone przez brzeg. Reprezentacja brzegowa to opis bryły składający się z dwóch części:

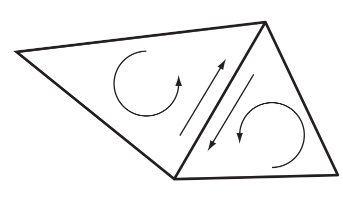
1. topologiczny opis połączeń i orientacji dla wierzchołków, krawędzi i ścian, np. przyleganie do siebie ścian

2. geometryczny opis umieszczenia tych elementów w przestrzeni, np. równania płaszczyzn zawierających ściany.

Zakładamy, że powierzchnia utworzona przez brzeg jest domkniętą, orientowalną rozmaitością 2-wymiarową, tzn. lokalnie homeomorficzną z R^2 (w dowolnym punkcie brzegu istnieje otoczenie homeomorficzne z dyskiem). Eliminuje to oczywiście brzegi utworzone przez powierzchnie stykające się w pojedynczych punktach lub na wspólnej krawędzi lub w ogóle przecinające się. Orientację (o ile istnieje) mówiąc ogólnie można sobie wyobrazić jako kierunek ruchu po zamkniętej krzywej wokół ustalonego punktu na powierzchni brzegowej obserwowanej z zewnątrz bryły (zgodnie lub przeciwnie do ruchu wskazówek zegara). Domknięte i orientowalne 2-rozmaitości dzielą przestrzeń na trzy opisane powyżej obszary: wnętrze, rozmaitość , zewnętrze. Niemniej boolowskie operacje regularne mogą dla dwóch rozmaitości dać wynik, który rozmaitością nie jest (łatwo sobie wyobrazić dwa sześciany o przekroju będącym fragmentem któreś ich krawędzi). To sugeruje trzy możliwe podejścia do problemu [2]:

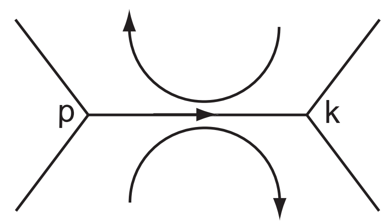
1. arbitralnie zabraniamy takich operacji, a jeśli wystąpią to modeler sygnalizuje błąd
2. interpretujemy wspólną krawędź jako dwie rozdzielne krawędzie
3. dopuszczamy taką sytuację jako normalną - podejście najprostsze i najczęściej stosowane.

Z racji, że najczęściej stosowaną reprezentacją brył jest przybliżanie ich za pomocą wielościanów, więc ograniczmy się do kompleksów symplicjalnych zamiast ogólnych rozmaitości. Przypomnijmy, że d-sympleks to brzeg powłoki wypukłej $d+1$ punktów w ustalonej przestrzeni. Jasne jest więc, że 0-sympleks to punkt, 1-sympleks jest odcinkiem, 2-sympleks trójkątem, a 3-sympleks czworościanem . Jasne jest też intuicyjnie, że d jest wymiarem d-sympleksu. Kompleks symplicjalny to rodzina sympleksów taka , że wraz z każdym sympleksem należy do niego każda ściana tego sympleksu oraz dowolne dwa sympleksy z rodziny są bądź rozłączne, bądź mają wspólną ścianę. Orientacja 2-sympleksu, czyli trójkąta może być określona na dwa sposoby poprzez ustalenie kolejności jego punktów, tzn. p_1, p_2, p_3 lub p_1, p_3, p_2. Orientacja 1-sympleksu czyli krawędzi trójkąta jest indukowana przez orientację 2-sympleksu. W przypadku kompleksu symplicjalnego orientację musimy ustalić tak, aby była ona *zgodna*, co oznacza w praktyce, że jeśli T_1 i T_2 są dwoma 2-sympleksami należącymi do kompleksu K, posiadającymi wspólną krawędź S to orientacje 1-sympleksu S indukowane odpowiednio przez T_1 i T_2 są przeciwne (rys. 2.11) Wtedy



Rysunek 2.11: Orientacje trójkątów o wspólnej krawędzi.

dla jednoznacznej reprezentacji wielościanu w R^3 potrzebujemy trzy tablice, po jednej dla wierzchołków , krawędzi i ścian. Ponieważ ściana jest określona przez uporządkowany zbiór krawędzi (określa to orientację ściany a także orientację krawędzi), więc w tablicy ścian umieścimy dla każdej z nich zbiór tego typu (lub lepiej zbiór wskaźników do krawędzi). Wierzchołek jest reprezentowany przez którąś z krawędzi do której należy, zatem w tablicy wierzchołków umieszczamy wskazanie do jednej z krawędzi zawierających go. W tablicy krawędzi dla każdej z nich będą umieszczone dwa jej wierzchołki (w ustalonej zgodnej z orientacją kolejności), lewa i prawa ściana przylegająca do krawędzi (lewa oznacza, że jest po naszej lewej stronie jeśli poruszamy się zgodnie z orientacją krawędzi), krawędzie poprzedzająca ją i następująca po niej w kierunku zgodnym z orientacją oraz analogiczne krawędzie w kierunku przeciwnym do orientacji (rys. 2.12) Powyższe tabli-



Rysunek 2.12: Kolejność punktów końcowych krawędzi wyznacza jej orientację .

ce zapewniają opis topologiczny obiektu. Do opisu geometrycznego trzeba co najmniej włączyć współrzędne wierzchołków oraz równania płaszczyzn zawierających ściany, przy czym trzeba pamiętać by ich równania indukowały normalne zgodne z orientacją ustaloną w opisie topologicznym.

Operatory Eulera W pewnych sytuacjach (np. spowodowanych błędami wynikającymi z dokładności numerycznej lub przeprowadzenia boolowskich operacji) dopuszczalna poprawność topologiczna ulega zakłóceniu. Aby temu zapobiec lub naprawić to co zostało zepsute został opracowany zbiór tzw. operatorów Eulera. Ich idea opiera się na uogólnionej regule Eulera dla wielościanów (kompleksów symplicjalnych). Jeśli V, E, F oznaczają odpowiednio ilość wierzchołków, krawędzi i ścian w obiekcie oraz G jest ilością dziur w nim, a S oznacza ilość spójnych obszarów tworzących brzeg figury (dopuszczamy zamknięte dziury w środku), zaś L ilość łamanych (zamkniętych bądź nie oraz mogących być w szczególności punktami) stanowiących brzegi scian to reguła mówi, że prawdziwy jest wzór :

$$V - E + F - (L - F) - 2(S - G) = 0$$

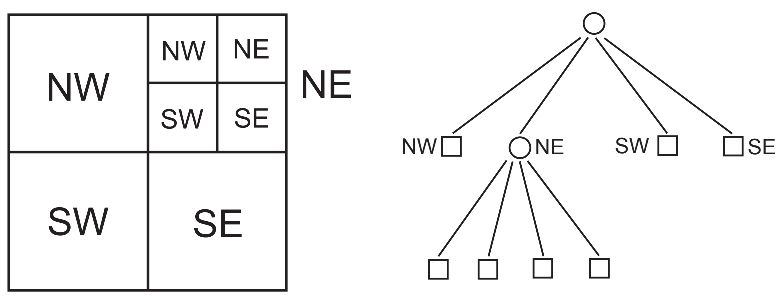
Operatory Eulera działają na bryłach spełniających tą regułę i w wyniku dają bryły które pomimo pewnych zmian tej własności nie tracą. Są to operatory typu mxky gdzie m pochodzi od *make*, k od *kill*, natomiast x reprezentuje elementy które operator może dodać do bryły, a y elementy mogące zostać usunięte. Przykładowo mekl dodaje krawędź i jednocześnie zmniejsza ilość separowalnych łamanych zamkniętych wyznaczających brzeg któreś ze ścian.

Operacje na reprezentacji brzegowej Podstawowe idee ich dotyczące jak i zarysowanie fragmentów algorytmów można znaleźć w [2].

2.2 Drzewa

2.2.1 Drzewa czwórkowe

Przedstawimy najpierw drzewa czwórkowe ze względu na fakt, iż po pierwsze posiadaja one własne zastosowania (np. przy meshing-u), a podrugie dlatego że stanowią one prostszą poglądowo wersję interesujących nas w modelowaniu bryłowym drzew ósemkowych. Przyjmijmy, że dany jest obszar kwadratowy zawierający zbiór n punktów. Przeprowadzimy procedurę podziału tego obszaru tak że w otrzymanych na końcu podobszarach będzie się znajdował co najwyżej jeden punkt. Procedura polega na rekurencyjnym podziale kwadratu - w każdym kroku dany kwadrat dzielimy na cztery równe podkwadraty. Oznaczmy je jako NW, NE, SW, SE. Ustawienie ich w strukturę drzew jest oczywiste (rys. 2.13) Jasne jest, że drzewo skonstruowane dla zbioru n punktów może mieć gałęzie



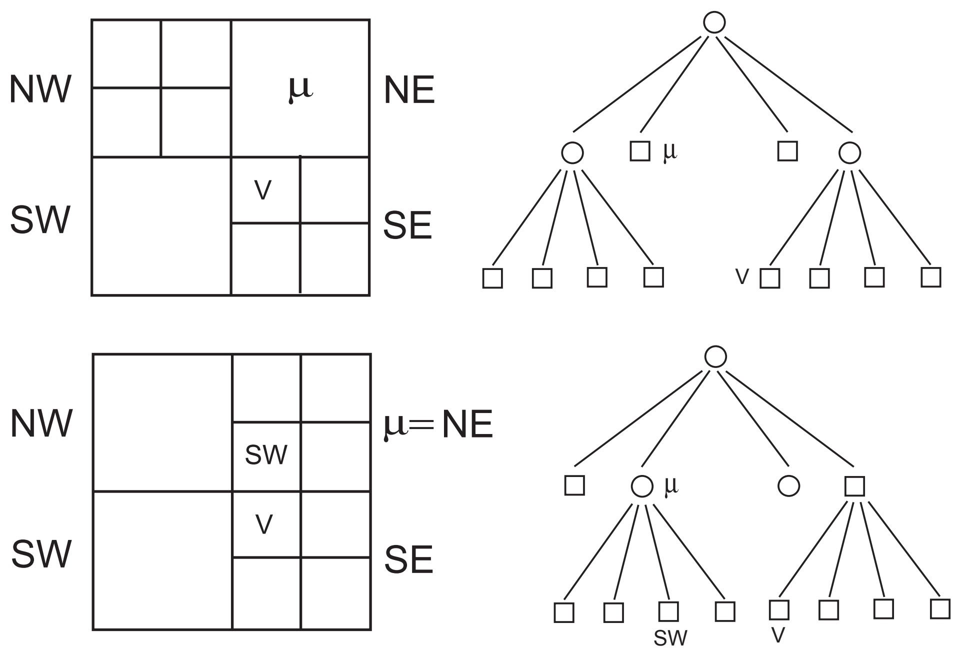
Rysunek 2.13: Drzewo czwórkowe.

różnej głębokości, tzn. drzewo może być np. całkowicie niezrównoważone. Trudne jest więc określenie rozmiaru drzewa i jego głebokości mając daną tylko ilość punktów. Niemniej łatwy dowód [1] pokazuje iż prawdziwy jest

Lemat 2.1 *Jeśli długość boku kwadratu inicjującego równa się s, a c jest najmniejszą odległością pomiędzy dwoma dowolnymi punktami ze zbioru P zawartego w kwadracie inicjującym, to głębokość otrzymanego drzewa czwórkowego wynosi co najwyżej $log(\frac{s}{c}) + \frac{3}{2}$*

Trzeba pamiętać, że w wypadku gdy punkty należą do krawędzi podziałów lub dokładnie do środka dzielonego kwadratu to musimy ustalić jednoznaczny ich przydział do podkwadratów. Ponadto można pokazać [1], że jeśli n punktów jest rozlokowanych w drzewie o głębokości d, to takie drzewo ma $O((d + 1)n)$ węzłów i jest konstruowane w czasie $O((d + 1)n)$.

Znajdowanie sąsiada w drzewie czwórkowym Oznaczmy przez $\sigma(v)$ kwadrat, odpowiadający węzłowi v (wewnętrznemu lub liściowi). Zadanie polega na znalezieniu bezpośredniego sąsiada dla $\sigma(v)$ w ustalonym kierunku N, E, S lub W, tzn. odnalezieniu węzła v' takiego, że $\sigma(v')$ bezpośrednio przylega do $\sigma(v)$ w danym kierunku. Dodatkowo nałóżmy jeszcze warunek, że v i v' są na tym samym poziomie głębokości drzewa, a jeśli nie ma takiego v' to szukamy najgłębszego węzła, którego kwadrat przylega do $\sigma(v)$. Jeśli z kolei i taki węzeł nie istnieje (zachodzi tylko w przypadeku, gdy $\sigma(v)$ ma krawędź zawartą w kwadracie inicjującym) to algorytm zwraca nil. Dla ustalenia uwagi przyjmijmy, że szukamy N sąsiada węzła v. Jeśli v jest SE lub SW potomkiem jakiegoś węzła to oczywiście rozwiązaniem będzie NE lub NW potomek tego samego drzewa. Jeśli v jest NE lub NW potomkiem to znajdujemy N sąsiada μ dla węzła rodzicielskiego elementu v. Jeśli μ jest węzłem wewnętrznym to N sąsiad v jest potomkiem μ, jeśli μ jest liściem wtedy szukanym węzłem jest μ. (rys. 2.14) . Algorytm ten można przedstawić następująco



Rysunek 2.14: (a) rodzicem dla v jest SE, więc N sąsiadem dla SE jest $\mu = NE$; μ jest liściem zatem on jest rozwiązaniem. (b) sytuacja podobna jak w (a), z tym że μ nie jest liściem, więc rozwiązaniem jest jego potomek SW.

[1]:

Algorytm 2.1 *Znajdowanie N sąsiada w drzewie czwórkowym*

```
N_sasiad(v,T)
{
  if (v == root(T)) return nil;
```

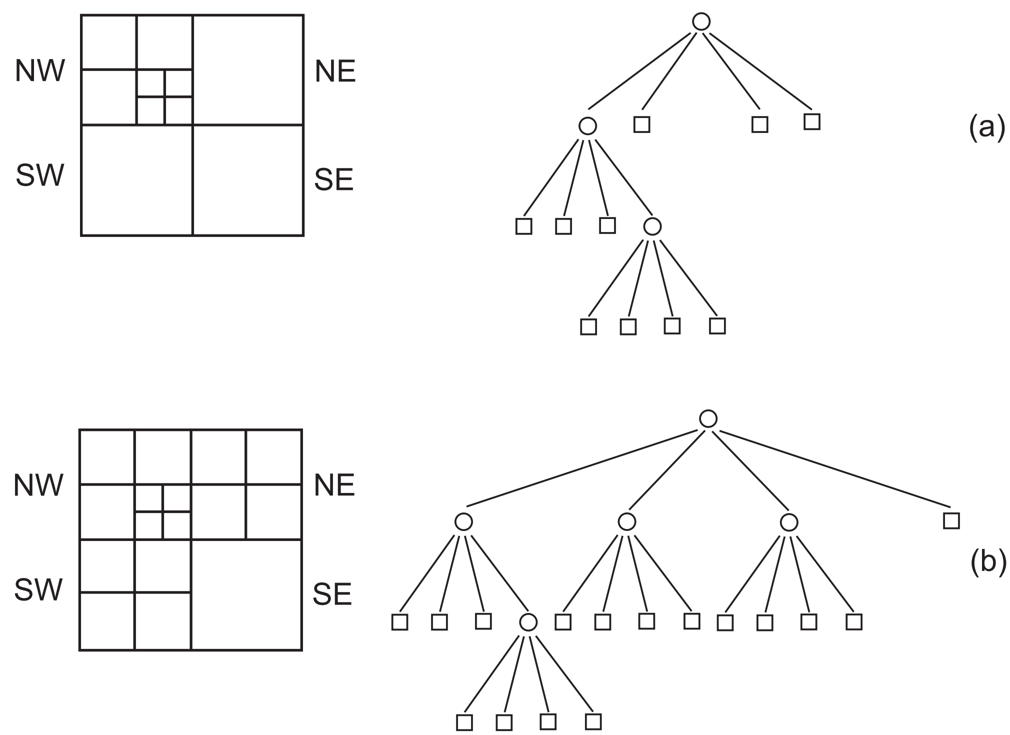
```
  if (v == SW_child(v) dla parent(v) ) return NW_child dla parent(v);
  if (v == SE_child(v) dla parent(v) ) return NE_child dla parent(v);
  m = N_sasiad(parent(v),T);
  if (m == nil) or (m == lisc) return m;
  else
    if (v == NW_child dla parent(v)) return SW_child dla m;
    else return SE_child dla m;
}
```

Nietrudno zauważyć, że dla drzewa głębokości d czas działania powyższego algorytmu jest rzędu $O(d+1)$

Drzewo czwórkowe zrównoważone W niektórych zastosowaniach drzewa niezrównoważone zachowują się źle. Przypomnijmy, że drzewo czwórkowe jest zrównoważone jeśli dwa sąsiadujące z sobą kwadraty odpowiadające jego węzłom różnią się długością krawędzi co najwyżej dwa razy. Oznacza to że dla danego drzewa czwórkowego niektóre z odpowiadających jego węzłom kwadraty trzeba będzie podzielić. Kryterium dzielenia będzie róznica długości krawędzi danego kwadratu w stosunku do jego sąsiadów przekraczająca czynnik 2. W języku węzłów drzewa oznacza to, że procedura znajdująca sąsiada w kierunku N zwraca węzeł, który ma SE lub SW potomka nie będącego liściem (rys. 2.15). Ponadto może się zdarzyć, że po podziale danego kwadratu, jego relacja w stosunku do innych jego sąsiadów zmieni się tak, iż któryś z nich będzie wymagał podziału. Szkic algorytmu dla drzewa zrównoważonego jest następujący [1]:

Algorytm 2.2 *Drzewo zrownoważone*

```
Drzewo_zrownowazone(T)
{
  Ustaw wszystkie liscie drzewa T w liniowa liste L;
  while (L != zbior pusty)
   {
     usun lisc m z listy L;
     if (kwadrat(m) wymaga podzialu)
       {
         podziel kwadrat(m) na cztery podkwadraty;
         if ( kwadrat(m) zawieral punkt P )
           przydziel punkt P do ktoregos z potomkow m;
         dodaj do L cztery nowe liscie ;
         if (kwadrat(m) posiada sasiadow wymagajacych podzialu)
           dodaj ich do L;
       }
   }
  return T;
}
```



Rysunek 2.15: (a) drzewo niezrównoważone, gdyż N sąsiad dla v ma SE potomka, który nie jest liściem - trzeba wykonać podział v. (b) drzewo po zrównoważeniu.

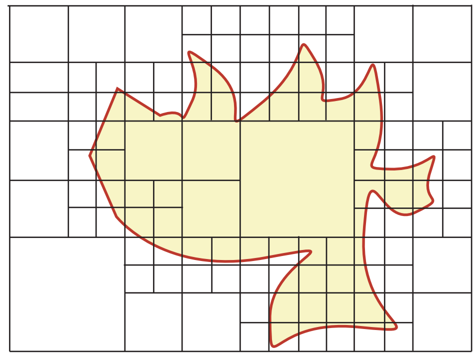
Reprezentacja figur płaskich przy pomocy drzew czwórkowych Daną figurę domkniętą, zawartą w inicjującym kwadracie pokrywamy coraz mniejszymi kwadratami posługując się podziałem stosowanym w konstrukcji drzewa czwórkowego. Kryterium zakończenia podziałów jest bądź osiągnięcie dopuszczalnej głębokości drzewa (tzn. długości krawędzi podkwadratów), bądź zajście sytuacji, gdy cały kwadrat jest wewnątrz lub na zewnątrz figury (rys. 2.16)

2.2.2 Drzewa ósemkowe

Zagadnienie jest 3-wymiarową wersją drzew czwórkowych i polega na podziale sześcianu na osiem równych kostek. Napisanie analogonów powyższych procedur nie powinno sprawić czytelnikowi problemu.

2.2.3 Drzewa PM

Drzewa te są wersją drzew ósemkowych, ale rozszerzają ilość kryteriów zakończenia podziałów z dwóch do pięciu. Obok liści całkowicie zawartych w bryle i liści całkowicie leżących poza nią dopuszczamy liście zawierające pojedynczy wierzchołek z brzegu bryły



Rysunek 2.16: Pokrycie figury kwadratami tworzącymi drzewo czwórkowe.

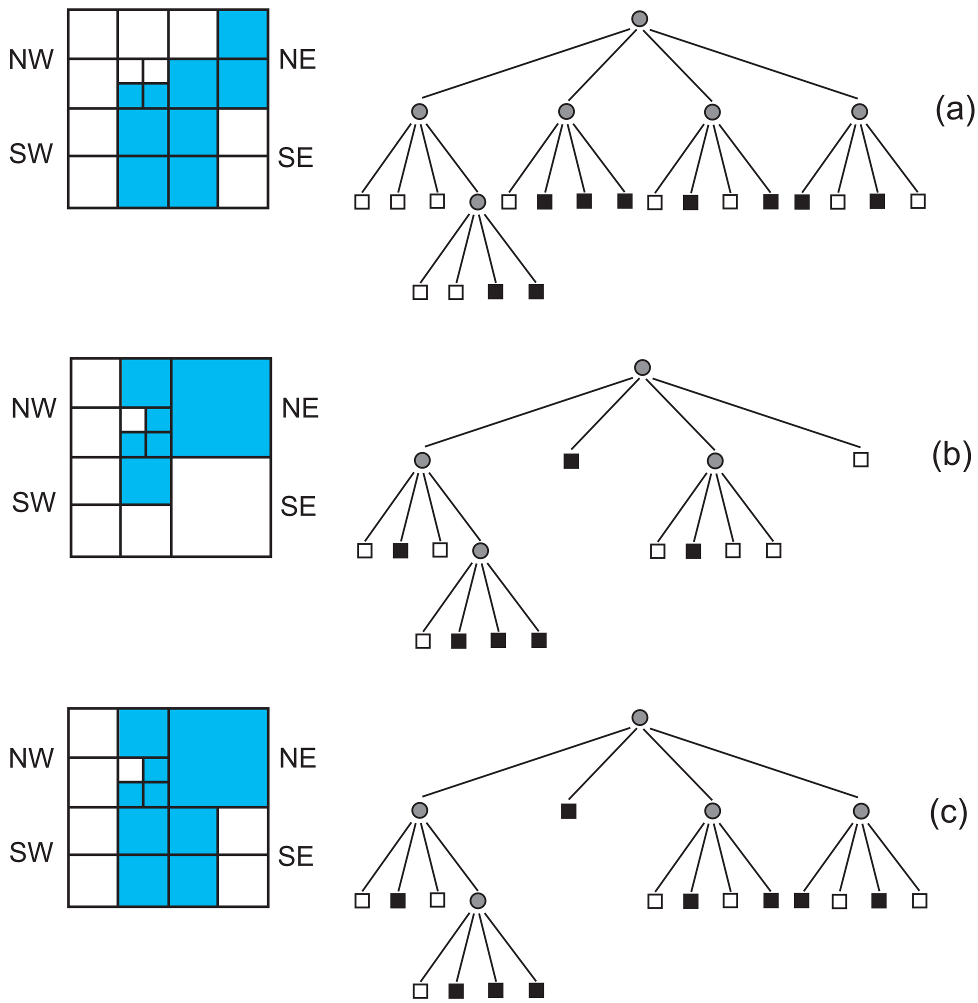
wraz z fragmentami wychodzących z niego krawędzi i ścian, liście zawierające fragment pojedynczej krawędzi i ścian do niej przylegających oraz liście zawierające fragment pojedynczej ściany bryły. Widać, że drzewa takie kombinują typowe drzewa ósemkowe z reprezentacją brzegową.

2.2.4 Drzewa BSP

Idea wynika z opisanej w wykładzie 2 konstrukcji drzew w których przestrzeń jest dzielona na rozłączne obszary separowalne przez brzegi półprzestrzeni zawierających ściany obiektu. Wtedy obiekt jest reprezentowany jako suma części określonych przez przekroje odpowiednich półprzestrzeni (zob. wykład 2 i ćwiczenia po nim).

2.2.5 Operacje boolowskie na drzewach typu octree

Możliwość wykonywania podstawowych operacji mnogościowych na bryłach reprezentowanych przy pomocy drzew jest jedną z podstawowych zalet takiego przedstawienia. Dla jasności pokażemy to w przypadku dwuwymiarowym, tzn. drzew czwórkowych. Załóżmy, że figury A i B mają taką reprezentację (rys. 2.17). Liscie czarne oznaczają kwadrat całkowicie zawarty w figurze, białe natomiast kwadrat będący całkowicie poza figurą. Węzły wewnętrzne są oznaczone kolorem szarym. Wtedy operacja sumy $A \cup B$ może być wykonana przez sumowanie wartości odpowiadających sobie wezłów w obu drzewach, przy czym działania są następujące: dla liści: czarny $\cup$ czarny = czarny, czarny $\cup$ biały = czarny, biały $\cup$ czarny = czarny, biały $\cup$ biały = biały oraz dla węzłów wewnętrznych: szary$\cup$szary=szary. W tym ostatnim przypadku wywołujemy procedurę sumującą rekurencyjnie dla potomków węzła wewnętrznego oraz po jej wykonaniu sprawdzamy czy przypadkiem wartości wszystkich potomków tego węzła w sumie nie są równe. Wtedy usuwamy je z drzewa zmieniając węzeł wewnętrzny na liść. Jako ćwiczenie dla czytelnika pozostawia się znalezienie analogicznej procedury dla przekroju.



Rysunek 2.17: (a) A, (b) B, (c) $A \cup B$

3 Triangulacje

3.1 Triangulacja wielokątów monotonicznych

Zostanie omówiona na ćwiczeniach. Dokładny jej opis można znaleźć w [1].

3.2 Triangulacja Delaunay

Triangulacja ta została opracowana z myślą o zastosowaniu jej w metodach reprezentacji terenu. Niech $f : A \to R$, gdzie $A \subset R^2$, będzie funkcją reprezentującą powierzchnię w R^3, której wartość $f(p)$ oznacza wysokość terenu w określonym punkcie P z dziedziny A. Wartość funkcji f jest dokładnie znana tylko w dyskretnym, skończonym podzbiorze $P \subset A$, w pozostałych zaś punktach jest interpolowana. Gdyby zastosować zamiast interpolacji diagramy Voronoi [1] dla A, generowane przez P to otrzymalibyśmy powierzchnię na kształt dwuwymiarowej funcji prostej (tzn. z uskokami). Zamiast tego dość niezręcznego rozwiązania przeprowadzimy traiangulację zbioru P i na jej bazie oszacujemy powierzchnię f. Problem sprowadza się więc do triangulacji zbioru punktów na płaszczyźnie. Analiza kształtów terenu pokazuje , że nienaturalne odwzorowanie możemy otrzymać, gdy w trójkątach będą występowały małe kąty, które odpowiadają połączeniu punktów odległych od siebie (może to powodować powstawanie nienaturalnych grani lub osi dolin, których w rzeczywistości nie ma). Skończona liczba punktów w P implikuje skończoną ilość możliwych triangulacji, zatem na pewno wśród nich istnieje taka, która maksymalizuje najmniejszy występujący w niej kąt. Właśnie to będzie kryterium, którym będziemy się posługiwać. Używając języka grafów zadanie triangulacji możemy przedstawić jako znalezienie *maksymalnego planarnego podziału A* wierzchołkami z P, tzn. takiego, że dodanie do niego chociażby jednej krawędzi zniszczy jego planarność (rozważamy tylko krawędzie będące odcinkami). Ponieważ można wykonać triangulację dowolnego wielokąta, więc jest jasne, że obszary pomiędzy wierzchołkami i krawędziami są trójkątami. Ponadto równie łatwo widać, że suma wszystkich trójkątów musi dać powłokę wypukłą P. Jaka będzie liczba trójkątów ? Oczywiście w jakiś sposób będzie zależała od liczby punktów w P. Problem rozstrzyga następujące twierdzenie [1]

Twierdzenie 3.1 *Jeśli P jest zbiorem n punktów na płaszczyźnie, z których nie wszystkie są współliniowe, k oznacza liczbę punktów leżących na brzegu powłoki wypukłej zbioru P, to każda triangulacja P zawiera 2(n-1)-k trójkątów i 3(n-1)-k krawędzi.*

Interesować nas będzie triangulacja minimalizująca kąty, ale specjalnego rodzaju. Rozpocznijmy od diagramu Voronoi dla P, czyli $Vor(P)$. Niech $\mathcal{G}$ oznacza dualny graf do $Vor(P)$, który z definicji ma węzeł w każdej komórce $Vor(P)$ i łuk łączący jego węzły odpowiadający krawędzi dzielącej dwie komórki. Istnieje jednoznaczna odpowiedniość pomiędzy ograniczonymi obszarami $\mathcal{G}$ i wierzchołkami $Vor(P)$. Narzucamy ponadto warunek na geometryczny graf $\mathcal{G}$, a mianowicie żądamy aby łuki w $\mathcal{G}$ były odcinkami. Wtedy ten geometryczny graf nazywamy *grafem Delaunay dla P* i oznaczamy $\mathcal{DG}(P)$ Graf ten ma kilka interesujących własności:

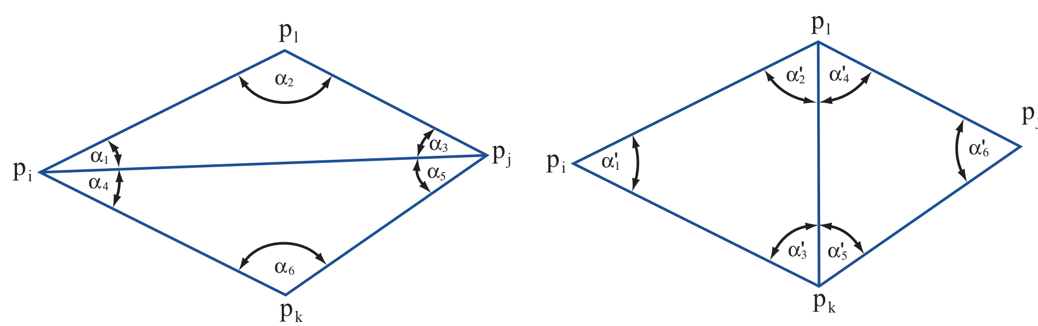
1. $\mathcal{DG}(P)$ jest grafem płaskim, gdy P jest zbiorem punktów na płaszczyźnie

2. trzy punkty $p_i, p_j, p_k \in P$ są wierzchołkami tego samego obszaru $\mathcal{DG}(P)$ wtedy i tylko wtedy, gdy koło, do którego brzegu należą, nie zawiera innych punktów z P w swoim wnętrzu,

3. dwa punkty p_i, p_j tworzą krawędź $\mathcal{DG}(P)$ wtedy i tylko wtedy, gdy istnieje domknięte koło C takie, że $p_i, p_j \in brzeg(C)$ oraz dla każdego $p \in P$, $p \neq p_i$, $p \neq p_j$ zachodzi $p \notin Int(C)$.

[1] Niech $P = \{p_1, \ldots, p_n\}$ będzie zbiorem n różnych punktów na płaszczyźnie. Wtedy *diagram Voronoi* jest definiowany jako podział płaszczyzny na n komórek, jedną dla każdego punktu p_i, przy czym dowolny punkt q leży w komórce zawierającej punkt p_i wtedy i tylko wtedy, gdy $dist(q, p_i) < dist(q, p_j)$ dla każdego $p_j \in P$, $j \neq i$.

Ponadto wiadomo, że jeśli v jest wierzchołkiem $Vor(P)$ należącym do k komórek, to odpowiadający mu obszar $f \in \mathcal{DG}(P)$ jest k-kątem wypukłym. Jeśli w P nie ma punktów takich, że cztery z nich leżą do brzegu jednego koła, to wszystkie wierzchołki $Vor(P)$ należą do trzech komórek, więc wszystkie komórki $f \in \mathcal{DG}(P)$ są trójkątami. *Triangulacją Delanuay* $\mathcal{TD}(P)$ nazywamy takie uzupełnienie geometrycznego grafu $\mathcal{DG}(P)$ krawędziami, iż wszystkie jego komórki są trójkątami. Wspomnieliśmy, że potrzebujemy znaleźć triangulację maksymalizującą najmniejszy kąt wśród jej elementów. Zanim wyjaśnimy jak żądanie to ma się do triangulacji Delanuay wprowadzimy kilka dodatkowych pojęć dla uściślenia zagadnienia. Niech $e = \overline{p_i p_j}$ będzie krawędzią $\mathcal{TD}(P)$. Jeśli e jest krawędzią wewnętrznego obszaru (tzn. ograniczonego) $\mathcal{TD}(P)$ to należy do dwóch trójkątów $p_i p_j p_k$ oraz $p_i p_j p_l$ (rys. 3.1) Usuwając krawędź $\overline{p_i p_j}$ oraz wprowadzając nową krawędź $\overline{p_l p_k}$



Rysunek 3.1: Kryterium zarządzającym triangulacją Delanuay jest maksymalizacja kątów w budowanych trójkątach.

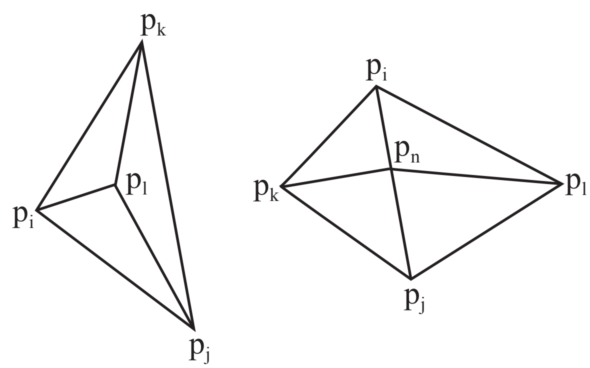
otrzymujemy inną triangulację. Jeśli

$$min\{\alpha_i :\ 1 \leqslant i \leqslant 6\} < min\{\alpha_i^{'} :\ 1 \leqslant i \leqslant 6\}$$

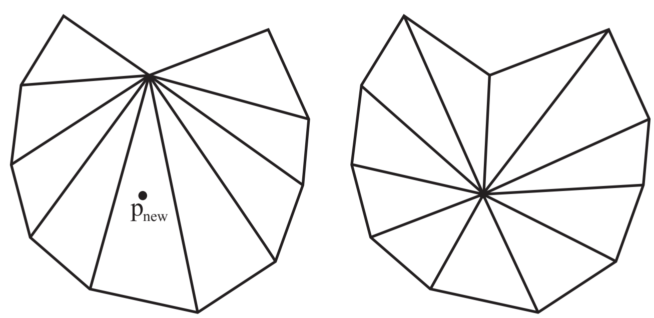
to mówimy , że krawędź $\overline{p_i p_j}$ jest *nielegalna*, natomiast krawędź $\overline{p_l p_k}$ *legalna*. Triangulację nazywamy *legalną*, jeśli nie zawiera nielegalnych krawędzi. Można dowieść [1], że

Twierdzenie 3.2 *Jeśli P jest zbiorem punktów na płaszczyźnie to $\mathcal{T}(P)$ jest legalna iff $\mathcal{T}(P) = \mathcal{TD}(P)$.*

Zatem triangulacja Delanuay jest dokładnie tą, której szukamy. Algorytm znajdujący $\mathcal{TD}(P)$, którym się posłużymy nie będzie korzystał ze znalezienia $\mathcal{DG}(P)$ dla $Vor(P)$ i następnie jego triangulacji, ale będzie polegał na znalezieniu triangulacji legalnej na coraz większym zbiorze punktów ze zbioru P tak długo, aż wszystkie z nich zostaną wzięte pod uwagę. Ściśle mówiąc będziemy triangulować zbiór $P \cup \{p_{-1}, p_{-1}, p_{-3}\}$, gdzie p_{-1}, p_{-1}, p_{-3} są takimi punktami, że $P \subset Int(\triangle(p_{-1}p_{-2}p_{-3}))$ oraz wystarczająco odległymi od P tak że końcowe usunięcie krawędzi zawierających punkty p_{-1}, p_{-2}, p_{-3} nie zniszczy zbudowanej triangulacji (w szczególności żaden z p_{-1}, p_{-2}, p_{-3} nie może należeć do okręgu zawierającego trzy punkty z P - por. własność 2.) Rozpoczynamy więc od $\triangle(p_{-1}p_{-2}p_{-3})$. Procedura postępowania będzie polegała na losowym dodawaniu kolejnych punktów z P i budowaniu legalnej triangulacji tak powiększszanego zbioru. Dodawany punkt $p_{new} \in P$ może leżeć bądź we wnętrzu jakiegoś trójkąta należącego do już zbudowanej triangulacji, bądź też należeć do krawędzi któregoś z trójkątów (rys. 3.2) Rysunek pokazuje jak będziemy



Rysunek 3.2: Nowy punkt leży w trójkącie (lewy przypadek) lub na jego krawędzi (prawy przypadek).



Rysunek 3.3: Czasami dodanie nowego punktu wymaga przebudowania całej dotychczasowej struktury trójkątów.

wtedy dodawać nowe krawędzie. Właściwie nie byłby żadnego problemu, gdyby nie fakt, iż dodanie nowych krawędzi może zniszczyć legalność istniejących wcześniej. W związku z tym po każdym dodaniu trzeba sprawdzić i ewentualnie naprawić tę właściwość dla wszystkich pontencjalnie mogących ją stracić krawędzi. Ponieważ naprawa jednych może zniszczyć z kolei inne, więc procedura naprawiająca ma typowo rekurencyjny charakter. W szczególności może dotyczyć wielu sąsiadujących z sobą trójkątów (rys. 3.3).

Procedura legalizacyjna może wyglądać następująco [1]:

```
LegalizeEdge(p_r,pi_pj,T) { pi_pj jest krawedzia ktora byc moze trzeba
                              bedzie usunac i zamienic na inna }
{
  if (pi_pj jest nielegalna) {pi_pj jest wspolna dla trojkatow pi_pj_pk
                              oraz pi_pj_pr}
    zastap pi_pj przez pr_pk;
    LegalizeEdge(pr,pi_pk,T);
    LegalizeEdge(pr,pk_pj,T);
}
```

cały zaś algorytm może mieć postać :

```
(0) T:=zbior pusty
(1) Wybierz odpowiednio trzy punkty p-1, p-2, p-3
(2) T:= T + trojkat(p-1,p-2,p-3)
(3) Oblicz losowa permutacje punktow p1,..,pn
(4) for {new=1,new<=n,new++}
     {
(5)    znajdz trojkat(pi,pj,pk) taki ze p_new nalezy do trojkat(pi,pj,pk)
(6)    if (p_new nalezy wnetrze(trojkat(pi,pj,pk)))
        {
(7)       dodaj_krawedzie(pi_p_new, pj_p_new, pk_p_new)
(8)       LegalizeEdge(p_new,pi_pj,T)
(9)       LegalizeEdge(p_new,pj_pk,T)
(10)      LegalizeEdge(p_new,pk_pi,T)
        }
(11)   else
        {
(12)      dodaj_krawedzie(p_new_pl, p_new_pk)
(13)      LegalizeEdge(p_new,pi_pl,T)
(14)      LegalizeEdge(p_new,pl_pj,T)
(15)      LegalizeEdge(p_new,pj_pk,T)
(16)      LegalizeEdge(p_new,pk_pi,T)
        }
     }
(17) usun punkty p-1, p-2, p-3 oraz wszystkie krawedzie je zawierajace.
```

Literatura

[1] Berg de, M., Krevald van, M., Overmars, M., Schwarzkopf, O., *Computational Geometry, Algorithms and Applications*, Springer Verlag, 1997,

[2] Hoffmann, C.M., *Geometric & solid modeling. An Introduction*, Morgan Kaufmann, 1989,

[3] Mortenson, M.E., *Geometric Modeling*, Wiley, 1997.