CHƯƠNG 6 :

MÔ HÌNH HÓA ĐỐI TƯỢNG 3 CHIỀU BẰNG LƯỚI ĐA GIÁC

Nội dung:

- Xây dựng công cụ làm việc với đối tượng trong không gian 3 chiều
- Biểu diễn khối rắn bằng lưới đa giác
- Vẽ mô hình khung dây đơn giản của đối tượng

TÓM TẮT

Trong chương này, chúng ta sẽ tìm hiểu cách mô tả đối tượng 3 chiều bằng lưới đa giác. Phần 6.1 trình bày khái quát quá trình mô hình hóa 3 chiều. Phần 6.2 trình bày về lưới đa giác. Sử dụng lưới đa giác, chúng ta có thể mô tả gần đúng bề mặt của những đối tượng 3 chiều phức tạp chỉ với cấu trúc dữ liệu đơn giản.

Phần 6.3 trình bày họ các khối rắn 3 chiều thú vị, chẳng hạn như khối Platonic, khối lăng trụ, khối vòm. Phần 6.4 nghiên cứu về họ các vật thể quét và trình bày cách tạo chữ và văn bản 3 chiều, ống, "hình con rắn" và mặt tròn xoay.

Phần 6.5 trình bày cách mô hình hóa khối rắn có bề mặt trơn bằng lưới đa giác . Lưới đa giác này chỉ là sự xấp xỉ bề mặt của khối rắn. Một công việc quan trọng là tính toán vector pháp tuyến của bề mặt tại một điểm bất kỳ. Chúng tôi sẽ trình bày một số họ khối rắn có bề mặt trơn, chẳng hạn như mặt bậc hai và siêu bậc hai, mặt tròn xoay, mặt chứa cạnh thẳng, mặt được biểu diễn bằng hàm tường minh của hai biến số.

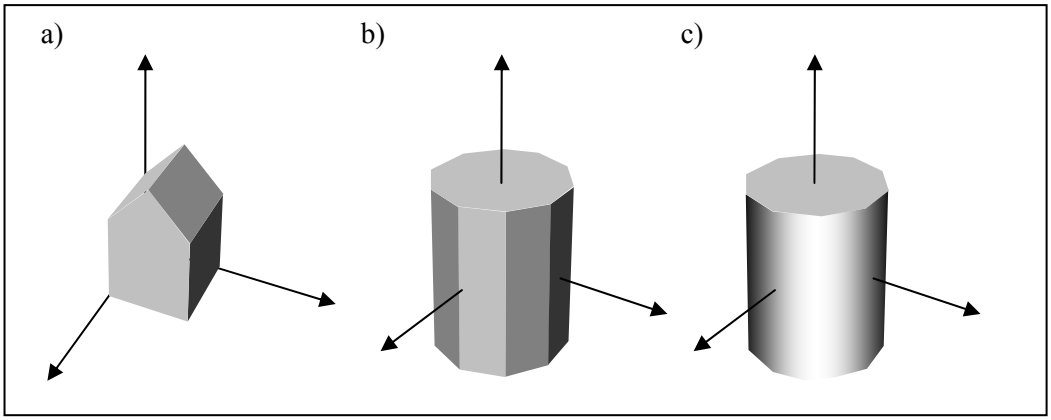
Phần thực hành ở cuối chương sẽ đi sâu hơn vào một số ý tưởng đã trình bày và yêu cầu bạn đọc tự xây dựng chương trình hiện thực các ý tưởng đó. Một số bài thực hành thiên về lý thuyết, chẳng hạn như tính toán vector pháp tuyến bằng phương pháp Newell và khảo sát dạng đại số của mặt bậc hai. Một số bài thực hành khác thiên về thực hành nhiều hơn, chẳng hạn như đọc số liệu của lưới đa giác từ tập tin hay tạo vật thể 3 chiều như ống và khối vòm.

6.1 GIỚI THIỆU

Lưới đa giác (polygonal meshes) là tập hợp của các đa giác phẳng (hay còn gọi là mặt) tạo nên bề mặt của một đối tượng. Chúng là phương pháp chuẩn để biểu diễn các khối rắn trong đồ họa. Chúng ta đã được xem một số ví dụ về khối lập phương, khối hai mươi mặt đều (icosahedron), cũng như sự xấp xỉ của các đối tượng có mặt trơn như hình cầu, hình trụ và hình nón. Trong chương này, chúng tôi sẽ trình bày khá nhiều ví dụ. Lưới đa giác được sử dụng nhiều là bởi vì sự đơn giản trong việc sử dụng: nó có thể được biểu diễn (bởi tập các điểm), có thuộc tính đơn giản (mỗi mặt đa giác có một vector pháp tuyến duy nhất), và dễ vẽ (dùng hàm tô màu đa giác hoặc bằng cách ánh xạ texture vào đa giác).

Rất nhiều các hệ thống đồ họa, bao gồm cả OpenGL, vẽ đối tượng 3 chiều bằng cách vẽ một tập các đa giác phẳng. Mỗi mặt đa giác được gửi xuống đường ống đồ họa, trải qua rất nhiều phép biến đổi, cho đến khi phần còn lại sau khi được cắn xén được tô màu và hiển thị lên màn hình.

Chúng tôi sẽ trình bày cách thiết kế những vật thể 3D phức tạp bằng cách định nghĩa tập các mặt thích hợp cho chúng. Một số vật thể có thể được biểu diễn một cách chính xác bởi lưới đa giác, trong khi đó, các vật thể khác chỉ được biểu diễn một cách gần đúng. Ngôi nhà trong hình 6.1(a) được tạo bởi các mặt đa giác, nên chúng ta sẽ nhìn thấy đường thẳng giữa các mặt. Như chúng ta đã biết hình trụ vốn có bề mặt trơn. Nhưng trong hình 6.1(b) tính trơn của mặt trụ không thể biểu diễn được nếu chỉ dùng các mặt đa giác: chúng ta nhìn thấy rất rõ mỗi mặt cũng như đường thẳng giao tuyến giữa các mặt. Tuy nhiên, một số kỹ thuật tô bóng sẽ làm cho một lưới đa giác như thế trở nên trơn tru, hình 6.1(c). Chúng ta sẽ nghiên cứu về cách tô bóng này (có tên là Gouraud) ở chương 8.



HÌNH 6.1 Một số vật thể được mô hình hóa bằng lưới đa giác

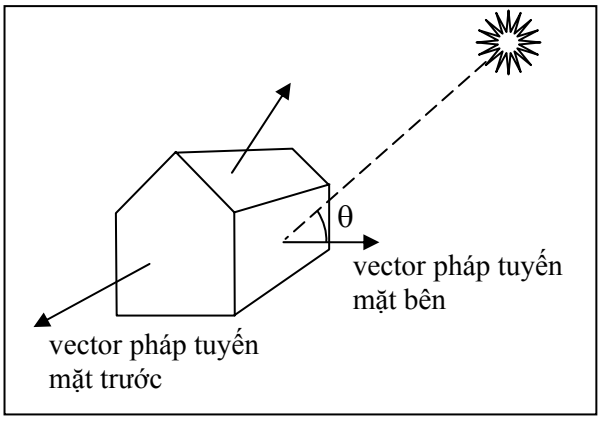
Bây giờ chúng tôi trình bày tổng quan về lưới đa giác, cách định nghĩa và xử lý chúng trong chương trình (có sử dụng thư viện OpenGL, hoặc không). Chúng ta sẽ áp dụng ý tưởng này để mô hình hóa khối đa diện (là những hình khối có các mặt là đa giác) và

nghiên cứu một vài họ các khối đa diện thú vị. Sau đó, chúng tôi sẽ trình bày cách dùng lưới đa giác để xấp xỉ những đối tượng 3 chiều có bề mặt cong và xây dựng những công cụ cần thiết để tạo và xử lý mô hình.

6.2 GIỚI THIỆU VỀ MÔ HÌNH HÓA KHỐI RẮN BẰNG LƯỚI ĐA GIÁC

Chúng ta dùng lưới đa giác để mô hình hóa cho cả khối rắn và những "bề mặt" mỏng. Một đối tượng được gọi là **khối rắn** (solid) nếu các mặt đa giác của nó xếp khít với nhau và đóng kín một phần không gian. Ngược lại, nếu các mặt không đóng kín một phần không gian, thì lưới đa giác được dùng biểu diễn một "bề mặt" mỏng. Trong cả hai trường hợp, chúng ta đều gọi tập các đa giác là **lưới đa giác** (polygonal mesh) (hoặc đơn giản là **lưới** (mesh)).

Lưới đa giác được biểu diễn bằng danh sách các đa giác, cùng với thông tin về hướng của mỗi đa giác. Thông tin về hướng này thường chỉ đơn giản là **vector pháp tuyến** (normal vector) của mặt phẳng chứa đa giác và được dùng trong quá trình tô bóng để xác định có bao nhiêu ánh sáng từ nguồn sáng được trải trên bề mặt. Hình 6.2 cho thấy vector pháp tuyến của các mặt khác nhau của ngôi nhà. Trong chương 8, chúng ta sẽ thấy độ sáng của bề mặt tỷ lệ với cosine của góc (như góc θ trong hình vẽ) giữa vector pháp tuyến của mặt với vector đến nguồn sáng. Vì thế, hướng của bề mặt so với nguồn sáng đóng vai trò quan trọng đối với màu sắc của bề mặt.

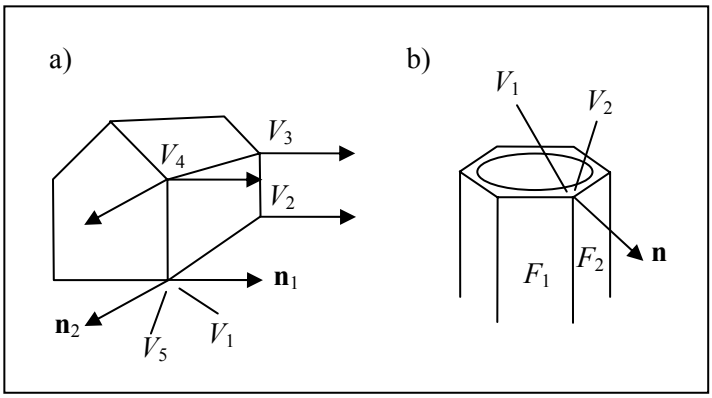


HÌNH 6.2 Hướng vector pháp tuyến của bề mặt quyết định độ sáng của bề mặt

Pháp tuyến đỉnh và pháp tuyến mặt

Rõ ràng rằng gán cho mỗi điểm của mặt một vector pháp tuyến sẽ có nhiều ích lợi và tiện dụng hơn so với dùng một vector pháp tuyến cho tất cả các điểm thuộc mặt. Như chúng ta sẽ thấy, cách làm này khiến cho quá trình cắt xén, cũng như quá trình tô bóng mặt cong trơn trở nên dễ dàng hơn. Đối với mặt phẳng ví dụ như bức tường của ngôi nhà, các

điểm V_1, V_2, V_3 và V_4 thuộc mặt bên ngôi nhà sẽ được gán cùng một vector pháp tuyến $\mathbf{n}_1$ (là vector pháp tuyến của mặt bên ngôi nhà), xem hình 6.3 (a). Nhưng điểm thuộc mặt trước, chẳng hạn như V_5, sẽ dùng vector pháp tuyến $\mathbf{n}_2$. (Lưu ý rằng hai điểm V_1 và V_5 thực ra là cùng một điểm trong không gian, nhưng chúng dùng vector pháp tuyến khác nhau).



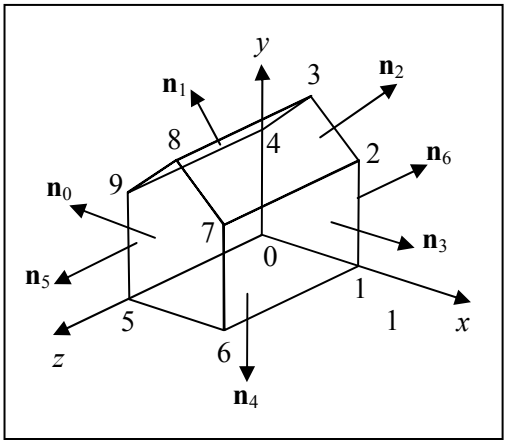
HÌNH 6.3 Gán pháp tuyến cho mỗi đỉnh thuộc mặt

Đối với mặt cong trơn, chẳng hạn như mặt trụ trong hình 6.3 (b), tuy vẫn được biểu diễn bằng lưới đa giác, nhưng bằng cách sử dụng cách tô bóng hợp lý, chúng ta hoàn toàn có thể tạo ra cảm giác trơn nhẵn cho bề mặt. Cả hai điểm V_1 thuộc mặt F_1 và V_2 thuộc mặt F_2 đều dùng chung vector pháp tuyến $\mathbf{n}$, vector này vuông góc với mặt cong tại điểm đang xét. Trong phần 6.2.2, chúng tôi sẽ trình bày cách tính vector pháp tuyến này.

6.2.1 ĐỊNH NGHĨA LƯỚI ĐA GIÁC

Lưới đa giác là tập các đa giác cùng với các vector pháp tuyến được gán cho mỗi đỉnh thuộc mỗi đa giác. Chúng ta sẽ bắt đầu bằng một ví dụ.

VÍ DỤ 6.2.1 Ngôi nhà



HÌNH 6.4 Hình ngôi nhà cơ bản

Hình 6.4 là một đối tượng đơn giản. Chúng ta gọi nó là ngôi nhà cơ bản. Nó có 7 mặt đa giác và 10 đỉnh (mỗi đỉnh đồng thời nằm trên 3 mặt). Để cho tiện lợi, giả sử sàn nhà là một hình vuông đơn vị. (Ngôi nhà sẽ được co dãn và định hướng một cách chính xác

trước khi đặt vào khung cảnh). Bởi vì, tường của ngôi nhà là phẳng, vì thế chỉ có bảy vector pháp tuyến, mỗi mặt của ngôi nhà sẽ có một vector pháp tuyến, như trong hình vẽ 6.4.

Có nhiều cách khác nhau để lưu trữ thông tin của lưới đa giác trong một tập tin hay trong chương trình. Đối với ngôi nhà trong ví dụ, chúng ta dùng một danh sách để lưu trữ 7 đa giác. Trong mỗi đa giác, chúng ta lại dùng một danh sách để lưu trữ vị trí cũng như vector pháp tuyến của mỗi đỉnh thuộc đa giác. (Có tất cả 30 đỉnh và 30 vector pháp tuyến). Nhưng cách tổ chức dữ liệu này tạo nên sự dư thừa và cồng kềnh, bởi vì thực ra chỉ có mười đỉnh khác nhau và 7 pháp tuyến khác nhau.

Một cách làm hiệu quả hơn là dùng 3 danh sách riêng biệt: một **danh sách đỉnh** (vertex list), một **danh sách pháp tuyến** (normal list) và một **danh sách mặt** (face list). Danh sách đỉnh chứa vị trí các đỉnh của lưới đa giác. Danh sách pháp tuyến chứa hướng các vector pháp tuyến. Còn danh sách mặt chỉ đơn giản chứa các chỉ mục của hai danh sách trên. Với cấu trúc dữ liệu này, chúng ta thấy ngôi nhà sẽ được mô tả bởi 10 đỉnh, 7 pháp tuyến và một danh sách đặc tả đơn giản cho 7 mặt.

Ba danh sách này làm việc cùng nhau: danh sách đỉnh chứa thông tin vị trí hay **thông tin hình học** (geometric information), danh sách pháp tuyến chứa **thông tin hướng** (orientation information) và danh sách mặt chứa thông tin liên kết hay **thông tin topology**.

đỉnh	x	y	z
0	0	0	0
1	1	0	0
2	1	1	0
3	0.5	1.5	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0.5	1.5	1
9	0	1	1

HÌNH 6.5 Danh sách đỉnh

pháp tuyến	n_x	n_y	n_z
0	-1	0	0
1	-.707	0.707	0
2	0.707	0.707	0
3	1	0	0
4	0	-1	0
5	0	0	1
6	0	0	-1

HÌNH 6.6 Danh sách vector pháp tuyến

Hình 6.5 là danh sách đỉnh của ngôi nhà. Hình 6.6 là danh sách 7 pháp tuyến. Các đỉnh có chỉ mục từ 0 đến 9, các pháp tuyến có chỉ mục từ 0 đến 6. Các vector pháp tuyến đã được chuẩn hóa, bởi vì hầu hết các thuật toán tô màu đều sử dụng vector đơn vị. (Nhớ lại rằng cosine có thể tính bằng tích vô hướng của hai vector đơn vị).

Hình 6.7 là danh sách mặt của ngôi nhà: mỗi mặt gồm danh sách các đỉnh và danh sách các vector pháp tuyến. Để tiết kiệm không gian lưu trữ, chúng ta chỉ sử dụng chỉ số của đỉnh và chỉ số của pháp tuyến. (Bởi vì các mặt của ngôi nhà là phẳng nên tất cả các đỉnh của mặt đều có cùng vector pháp tuyến). Danh sách các đỉnh của mỗi mặt sẽ được bắt đầu với bất kỳ đỉnh nào, sau đó chạy quanh mặt, mỗi đỉnh một lần cho đến khi các đỉnh được duyệt hết. Có hai cách duyệt các đỉnh của đa giác: theo chiều kim đồng hồ và ngược chiều kim đồng hồ. Ví dụ, mặt #5 có hai cách duyệt (5, 6, 7, 8, 9) hoặc (9, 8, 7, 6, 5). Có thể

sử dụng bất kỳ cách duyệt nào, nhưng chúng ta sẽ tuân thủ theo quy tắc dưới đây đã được chứng minh rằng có nhiều ưu điểm

Duyệt các đỉnh của đa giác ngược chiều kim đồng hồ khi chúng ta quan sát đối tượng từ bên ngoài.

Theo thứ tự này, nếu bạn duyệt các đỉnh của mặt, bằng cách đi trên đường biên của mặt từ đỉnh này đến đỉnh khác, thì phần trong của mặt sẽ nằm phía tay trái bạn. Sau này chúng ta sẽ thiết kế nhiều giải thuật để lợi dụng thứ tự này. Nhờ cách duyệt theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ, chúng ta có thể dễ dàng phân biệt được "mặt trước" hoặc "mặt sau" của mặt đa giác.

mặt	các đỉnh	các pháp tuyến
0 (trái)	0, 5, 9, 4	0, 0, 0, 0
1 (mái nhà trái)	3, 4, 9, 8	1, 1, 1, 1
2 (mái nhà phải)	2, 3, 8, 7	2, 2, 2, 2
3 (phải)	1, 2, 7, 6	3, 3, 3, 3
4 (đáy)	0, 1, 6, 5	4, 4, 4, 4
5 (trước)	5, 6, 7, 8, 9	5, 5, 5, 5, 5
6 (sau)	0, 4, 3, 2, 1	6, 6, 6, 6, 6

HÌNH 6.7 Danh sách mặt của ngôi nhà

Ngôi nhà là ví dụ về mô hình "hướng dữ liệu" (data – intensive), với tọa độ của mỗi đỉnh được nhập bởi người thiết kế. Ngược lại, sau này, chúng ta sẽ thấy một vài mô hình được tạo bởi các giải thuật. Chúng ta có thể dễ dàng tính toán được tọa độ các đỉnh của ngôi nhà: người thiết kế chọn một hình vuông đơn vị cho sàn nhà, đặt một góc nhà trùng với gốc tọa độ, chọn chiều cao của ngôi nhà là 1.5 đơn vị. Bằng cách sử dụng phép biến đổi tỷ lệ, có thể thay đổi những kích thước này (tuy vậy, tỷ lệ tương đối giữa chiều cao của bức tường và nóc nhà luôn có giá trị cố định là 1: 1.5).

6.2.2 TÌM VECTOR PHÁP TUYẾN

Chúng ta có thể nhập vị trí các đỉnh bằng tay, nhưng không dễ dàng chút nào khi tính toán vector pháp tuyến. Thông thường, mỗi mặt sẽ có từ 3 đỉnh trở lên, và rất khó khăn nếu như tính toán vector pháp tuyến bằng tay. Cách làm tốt nhất là để cho máy tính thực hiện công việc này trong quá trình tạo mô hình.

Nếu mặt là đa giác phẳng, chẳng hạn như trong trường hợp ngôi nhà, chúng ta chỉ cần tìm vector pháp tuyến của mặt đó và gán giá trị tìm được cho mỗi đỉnh thuộc mặt. Cách làm trực tiếp để tìm pháp tuyến là dùng tích có hướng, như hình 4.16. Chúng ta lấy 3 điểm kề nhau bất kỳ thuộc mặt, ví dụ V_1, V_2 và $V3$, rồi dùng công thức tính tích có hướng $\mathbf{m} = (V_1 - V_2) \times (V_3 - V_2)$ để tính toán pháp tuyến . Sau đó, chúng ta chuẩn hóa vector pháp tuyến này.

Có hai vấn đề nảy sinh nếu sử dụng cách tính toán đơn giản này. Thứ nhất, nếu hai vector $(V_1 - V_2)$ và $(V_3 - V_2)$ gần như song song với nhau, thì tích có hướng của chúng sẽ

rất nhỏ, và hạn chế về độ chính xác số thực của máy tính có thể sẽ làm cho giá trị này bằng không. Thứ hai, như sau này chúng ta sẽ thấy, một đa giác chưa hẳn đã phải là một đa giác phẳng thực sự, tức là, các đỉnh của nó không cùng nằm trên một mặt phẳng. Như thế, bề mặt được biểu diễn bởi các đỉnh này không thực sự phẳng hoàn toàn. Trong trường hợp này, chúng ta cần dùng tất cả các đỉnh để tính giá trị "trung bình" cho vector pháp tuyến.

Martin Newell đã đưa ra phương pháp có thể khắc phục được hai vấn đề kể trên. Phương pháp này tính toán tọa độ thành phần m_x, m_y và m_z của vector pháp tuyến **m** dựa trên công thức sau:

$$m_x = \sum_{i=0}^{N-1} (y_i - y_{next(i)})(z_i + z_{next(i)}),$$
$$m_y = \sum_{i=0}^{N-1} (z_i - z_{next(i)})(x_i + x_{next(i)}), \qquad (6.1)$$

và

$$m_z = \sum_{i=0}^{N-1} (x_i - x_{next(i)})(y_i + y_{next(i)})$$

với N là số đỉnh của mặt, (x_i, y_i, z_i) là tọa độ của đỉnh thứ i và next(j) = (j + 1) mod N là chỉ số của đỉnh "tiếp theo" sau đỉnh j, khi j nhận giá trị N-1 thì next(j) quay lại 0. Để tính toán mỗi tọa độ của vector pháp tuyến chúng ta chỉ cần thực hiện một phép nhân cho mỗi cạnh, và không cần phải kiểm tra tính đồng phẳng của các đỉnh. Vector **m** được tính toán bằng phương pháp Newell có thể chỉ vào phía ngoài hoặc phía trong của đa giác. Trong phần thực hành, chúng ta sẽ thấy, nếu các đỉnh của đa giác được duyệt theo chiều ngược kim đồng hồ (CCW) khi nhìn từ phía ngoài của đa giác, thì **m** sẽ hướng ra phía ngoài của mặt.

VÍ DỤ 6.2.2

Cho đa giác với các đỉnh là P_0 = (6, 1, 4), P_1 = (7, 0, 9) và P_2 = (1, 1, 2). Tìm pháp tuyến của đa giác bằng phương pháp Newell.

Lời giải:

Nếu áp dụng trực tiếp công thức tính tích có hướng, ta có **m** = ((7, 0, 9)- (6, 1, 4)) × ((1, 1, 2) – (6, 1, 4)) = (2, -23, -5). Sử dụng phương pháp Newell cũng cho cùng kết quả là (2, -23, -5).

BÀI TẬP

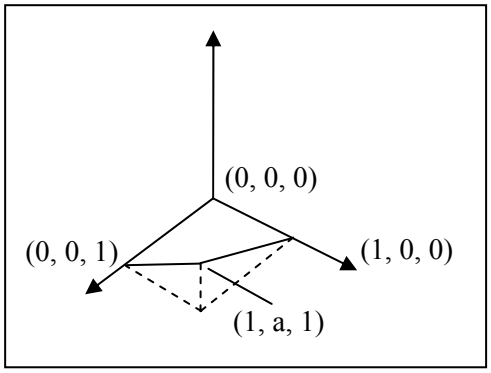
6.2.1 Sử dụng phương pháp Newell

Cho ba điểm (6, 1, 4), (2, 0, 5) và (7, 0, 9), hãy so sánh pháp tuyến tìm được bằng cách sử dụng phương pháp Newell và pháp tuyến tìm được bằng cách sử dụng công thức tính tích có hướng. Sau đó, dùng phương pháp Newell để tìm vector pháp tuyến **n** = (n_x, n_y, n_z) của đa giác có các đỉnh là (1, 1, 2), (2, 0, 5), (5, 1, 4) và

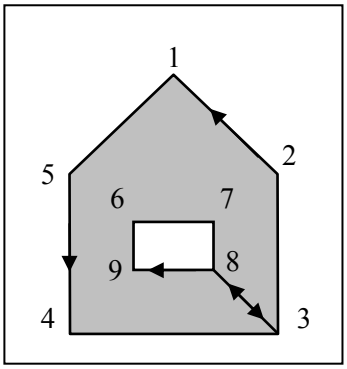
(6, 0, 7). Đây có phải là đa giác phẳng không? Nếu nó là đa giác phẳng, hãy tìm vector pháp tuyến của nó bằng cách sử dụng tích có hướng, và so sánh với kết quả với giá trị tìm được bằng phương pháp Newell.

6.2.2 Đa giác không phẳng

Cho tứ giác như hình 6.8 với các đỉnh là (0, 0, 0), (1, 0, 0), (0, 0, 1) và (1, *a*, 1). Khi *a* khác 0, tứ giác này không phải là một đa giác phẳng. Hãy tìm "pháp tuyến" của đa giác này bằng phương pháp Newell .



HÌNH 6.8 Một đa giác không phẳng



HÌNH 6.9 Mặt chứa một lỗ hổng

6.2.3 Biểu diễn hình lập phương

Tạo danh sách đỉnh, danh sách pháp tuyến và danh sách mặt cho một hình lập phương có tâm trùng với gốc tọa độ, có các cạnh song song với các trục tọa độ và có chiều dài bằng hai đơn vị. Tám đỉnh của hình lập phương nằm ở tọa độ $(\pm 1, \pm 1, \pm 1)$.

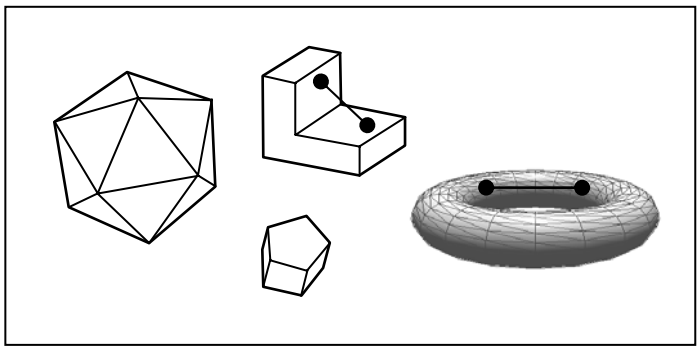
6.2.4 Mặt có chứa lỗ hổng

Hình 6.9 là một mặt có chứa lỗ hổng. Chúng ta thêm vào một cặp cạnh giả tưởng để nối đường biên của mặt với đường biên của lỗ hổng như trong hình vẽ.
Khi duyệt các đỉnh của mặt sao cho phần trong của mặt nằm bên tay trái, thì các đỉnh của lỗ hổng sẽ được duyệt theo chiều kim đồng hồ. Giả sử chúng ta nhìn vào mặt trong hình vẽ từ phía ngoài, thì các đỉnh sẽ được duyệt theo thứ tự sau: 5, 4, 3, 8, 9, 6, 7, 8, 3, 2, 1. Giả sử, ta thêm một lỗ hổng nữa, hãy vẽ mặt và danh sách các đỉnh của mặt. Vector pháp tuyến được gán cho mỗi đỉnh là gì?

6.2.3 TÍNH CHẤT CỦA LƯỚI ĐA GIÁC

Cho một lưới đa giác bao gồm danh sách đỉnh, danh sách pháp tuyến và danh sách mặt, chúng ta muốn biết đối tượng mà lưới đa giác này biểu diễn là gì. Sau đây là một số tính chất cần quan tâm:

- **Tính đặc** (solidity): lưới biểu diễn một khối rắn nếu các mặt của nó đóng kín một phần không gian hữu hạn.
- **Tính liên thông** (connectedness): lưới được gọi là liên thông nếu giữa hai đỉnh bất kỳ tồn tại một đường dẫn đi theo các cạnh nối. (Nếu lưới không liên thông, người ta thường biểu diễn nó bằng nhiều lưới khác nhau).
- **Tính đơn giản** (simplicity): lưới được gọi là đơn giản nếu nó biểu diễn một khối rắn và không có lỗ hổng bên trong.
- **Tính phẳng** (planarity): lưới được gọi là phẳng nếu mỗi mặt của nó là một đa giác phẳng: tức là, tất cả các đỉnh của một mặt đều nằm trên cùng một mặt phẳng. Một số giải thuật đồ họa sẽ làm việc hiệu quả hơn nếu mặt là đa giác phẳng. Tam giác đương nhiên là đa giác phẳng, nên một vài phần mềm đồ họa đã lợi dụng tính chất này bằng cách chỉ sử dụng tam giác. Trong khi đó, tứ giác có thể là phẳng hoặc không phẳng. Ví dụ, tứ giác trong hình 6.8 là phẳng nếu và chỉ nếu $a = 0$.
- **Tính lồi** (convexity): lưới biểu diễn đối tượng lồi nếu đoạn thẳng nối hai điểm bất kỳ của đối tượng nằm hoàn toàn bên trong đối tượng. Chúng tôi đã trình bày về tính lồi của đa giác trong phần 2.3.6. Hình 6.10 cho thấy một số đối tượng có tính lồi và một số đối tượng không có tính lồi. Với những đối tượng không có tính lồi, tồn tại ít nhất một đoạn thẳng có hai đầu mút thuộc đối tượng nhưng bản thân đoạn thẳng lại không nằm trọn trong đối tượng.

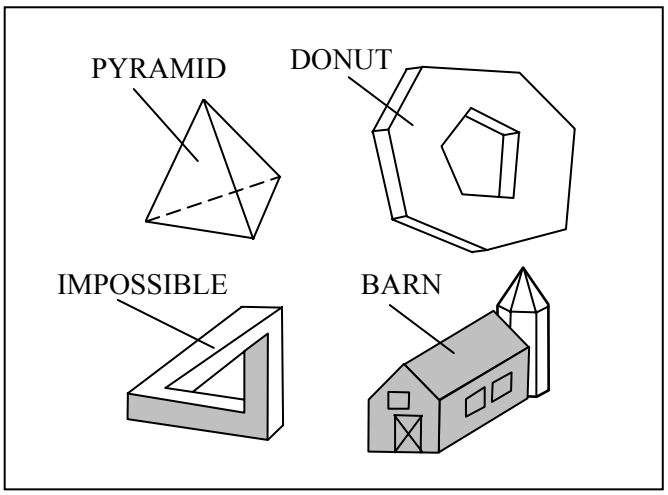


HÌNH 6.10 Ví dụ về một số đối tượng 3D lồi và không lồi

Ngôi nhà trong ví dụ 6.2.1 có đầy đủ những tính chất trên. (Hãy tự kiểm chứng). Với một lưới đa giác, chúng ta có thể dễ dàng kiểm chứng một số tính chất bằng chương trình dựa trên các thuật toán đơn giản. (Chúng ta sẽ thảo luận về những thuật toán này sau). Trong khi đó những tính chất khác, chẳng hạn như tính đặc, rất khó kiểm chứng.

Lưới đa giác chúng ta chọn để mô hình hóa đối tượng trong đồ họa có thể có một vài hoặc tất cả những tính chất kể trên: sự lựa chọn này sẽ phụ thuộc vào việc chúng ta sẽ sử dụng lưới đa giác để làm gì. Nếu lưới đa giác được dùng để biểu diễn một đối tượng vật lý được làm bằng vật liệu nào đó, thì khi muốn xác định được khối lượng và trọng tâm của nó, ít nhất lưới đa giác cũng phải thỏa mãn tính chất liên thông và tính đặc. Nếu chúng ta chỉ muốn vẽ đối tượng thì những ràng buộc sẽ được nới rộng, bởi vì rất nhiều đối tượng vẫn có thể được vẽ ra mặc dù chúng không tồn tại trong thực tế.

Hình 6.11 là ví dụ về một vài đối tượng mà chúng ta có thể muốn biểu diễn chúng bằng lưới đa giác. PARAMID được tạo bởi các mặt tam giác, và đương nhiên các tam giác này là các đa giác phẳng. PARAMID không chỉ có tính lồi, mà thực ra nó có tất cả các tính chất được nêu trên.



HÌNH 6.11 Ví dụ về một số khối rắn được miêu tả bằng lưới đa giác

DONUT liên thông và đặc, nhưng không lồi và không đơn giản (nó có chứa một lỗ hổng bên trong). Hai mặt trên và dưới của nó đều có chứa lỗ hổng. Chỉ dựa trên hình vẽ, không thể xác định các mặt của DONUT có phải là đa giác phẳng hay không. Sau này, chúng tôi sẽ đưa ra một thuật toán để xác định tính phẳng dựa vào danh sách mặt và danh sách đỉnh.

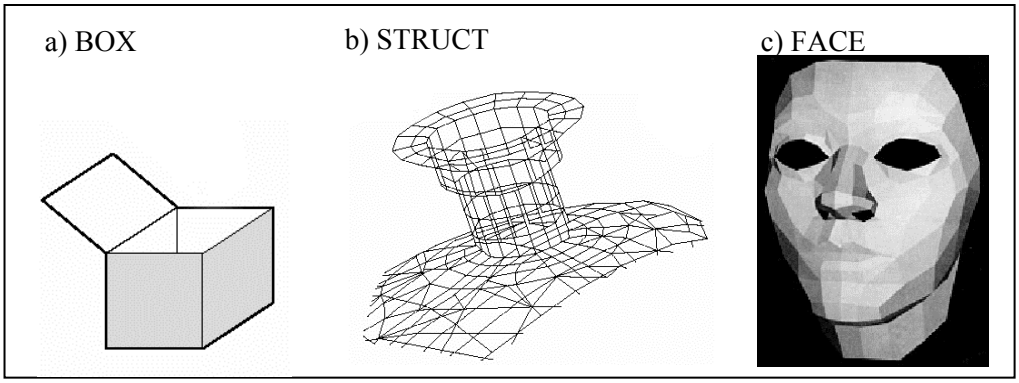
IMPOSSIBLE không tồn tại trong thực tế. Liệu nó có thể được biểu diễn bằng một lưới đa giác hay không?

BARN dường như gồm hai phần, nhưng nó có thể được biểu diễn chỉ bởi một lưới đa giác. Nó có liên thông hay không phụ thuộc vào việc các mặt ở chỗ tiếp giáp giữa ngôi nhà chính và nhà chứa thức ăn được định nghĩa như thế nào. BARN minh họa cho một trường hợp thường gặp trong đồ họa: một số mặt được thêm vào lưới đa giác, chẳng hạn như cửa sổ và cửa ra vào, giúp chúng ta có được không gian để tô vẽ texture lên một phần thuộc mặt. Trong ví dụ này, thực ra mặt bên của ngôi nhà là một hình chữ nhật không chứa bất kỳ lỗ hổng nào, ở đây hai hình chữ nhật được thêm vào để biểu diễn hai cửa sổ. Những hình chữ nhật này nằm trên cùng mặt phẳng của mặt bên ngôi nhà. Lưới đa giác này là không liên thông, nhưng nó vẫn có thể được hiển thị trên thiết bị đồ họa.

6.2.4 DÙNG LƯỚI ĐA GIÁC MÔ HÌNH HÓA ĐỐI TƯỢNG KHÔNG ĐẶC

Hình 6.12 là ví dụ của một vài đối tượng có thể được biểu diễn bằng lưới đa giác. Các đối tượng này là các bề mặt, và cách hình dung tốt nhất là coi chúng như là những "cái vỏ" (shell) có chiều dày vô cùng bé.

BOX là một chiếc hộp mở có nắp được nâng lên. Trong ngữ cảnh đồ họa, có thể chúng ta muốn tô màu xanh cho sáu mặt bên ngoài của BOX và màu đỏ cho các mặt bên trong. (Nếu chúng ta bỏ đi một mặt của PYRAMID trong hình 6.11, chúng ta sẽ nhận được gì?)



HÌNH 6.12 Một số mặt được mô tả bằng lưới đa giác

Trong hình vẽ còn có hai bề mặt phức tạp là STRUCT và FACE. Ở đây, chúng ta dùng lưới đa giác để xấp xỉ đối tượng có bề mặt cong trơn. Trong một số trường hợp, chúng ta dùng phương pháp số hóa các điểm trên mặt người để nhận được lưới đa giác biểu diễn đối tượng. Nếu mỗi mặt đa giác được tô bằng một màu, thì hình ảnh nhìn không được chân thực và trơn tru, như chúng ta thấy trong hình 6.12 (a). Sau này, chúng ta sẽ tìm hiểu cách vẽ bề mặt trơn chỉ cần dựa vào mô hình lưới đa giác của đối tượng.

Rất nhiều phần mềm xây dựng mô hình cho đối tượng (khối rắn hoặc bề mặt) cố gắng mô phỏng hình dạng thực của đối tượng bằng lưới đa giác. Vấn đề tạo danh sách đỉnh, danh sách pháp tuyến và danh sách mặt có thể rất khó khăn. Ví dụ, giả sử chúng ta phải viết một giải thuật tạo ra danh sách đỉnh và danh sách mặt để mô phỏng xấp xỉ hình dáng của một bộ phận cơ khí, hay một tòa nhà. Với số mặt đủ lớn, chúng ta có thể dùng lưới đa giác để mô phỏng một bề mặt nào đó với bất kỳ độ chính xác nào. Tính chất này khiến cho lưới đa giác trở thành công cụ được sử dụng rộng rãi trong mô hình hóa.

6.2.5 LÀM VIỆC VỚI LƯỚI ĐA GIÁC TRONG CHƯƠNG TRÌNH

Chúng ta muốn có một cách làm hiệu quả và dễ dàng trong chương trình để tạo và vẽ các đối tượng được biểu diễn bởi lưới đa giác. Dữ liệu của lưới đa giác thường được lưu trữ trong tập tin, nên chúng ta cũng cần có một phương pháp đơn giản để đọc và viết những tập tin này. Một cách làm tự nhiên là định nghĩa lớp *Mesh* và nhúng các chức năng mong muốn vào lớp này.

```
//################## VertexID ###################
class VertexID
{
     public:
              int   vertIndex; //index of this vertex in the vertex list
              int   normIndex; // index of this vertex's normal
```

```
};
//################### Face ###################
class Face
{
  public:
          int        nVerts;  // number of vertice in this face
          VertexID*  vert;  // the list of vertex and normal index
          Face() { nVerts = 0; vert = NULL; }
          ~Face() { delete[] vert; nVerts = 0; }
};
//################### Mesh ###################
class Mesh
{
  private:
          int         numVerts;  // number of vertices in the mesh
          Point3*     pt;     // array of 3D vertices
          int         numNormals;  // number of normal vectors for the mesh
          Vector3*    norm;  // array of normals
          int         numFaces;  // number of faces in the mesh
          Face*       face;  // array of face data
          // ... others to be added later
  public:
          Mesh();
          ~Mesh();
          int  readFile(char* fileName);
          // ... others
};
```

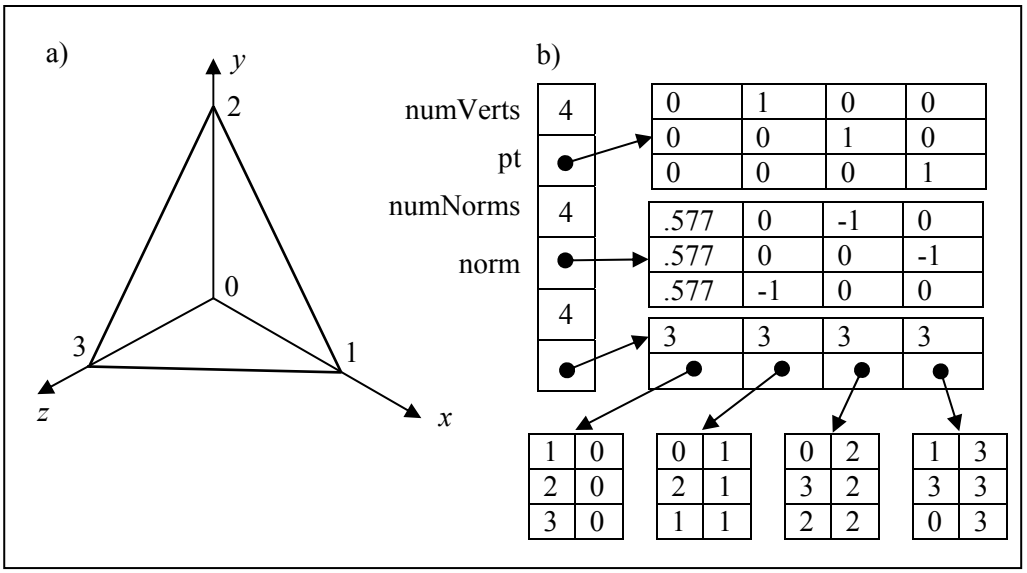
HÌNH 6.13 Dữ liệu đề nghị cho lưới đa giác

Hình 6.13 là phần khai báo của lớp `Mesh` cùng với hai lớp hỗ trợ là `VertexID` và `Face`. Đối tượng của lớp `Mesh` có danh sách đỉnh, danh sách pháp tuyến và danh sách mặt lần lượt được chứa trong các mảng `pt`, `norm` và `face`. Các mảng này được phân phối động trong thời gian chạy, khi chương trình biết được chúng phải có kích thước như thế nào. Chiều dài của chúng lần lượt được chứa trong các biến `numVerts`, `numNormals` và `numFaces`. Chúng ta có thể thêm vào những biến dữ liệu khác để mô tả các tính chất vật lý của đối tượng, chẳng hạn như khối lượng và loại vật liệu làm nên đối tượng.

Kiểu dữ liệu `Face` về cơ bản là danh sách các đỉnh và các vector pháp tuyến tương ứng với mỗi đỉnh thuộc mặt. Nó là mảng của bộ đôi các chỉ số: đỉnh thứ `v` trong mặt `f` có vị trí là `pt[face[f].vert[v].vertIndex]` và vector pháp tuyến tương ứng là `norm[face[f].vert[v].normIndex]`. Các tổ chức này thoạt nhìn có vẻ như cồng kềnh, nhưng cách đánh chỉ số này rất dễ quản lý và hiệu quả. Nó cho phép chúng ta truy xuất ngẫu nhiên nhanh chóng vào mảng `pt[]`.

VÍ DỤ 6.2.3 Dữ liệu của tứ diện

Hình 6.14 là hình vẽ tứ diện có các đỉnh là (0, 0, 0), (1, 0, 0), (0, 1, 0) và (0, 0, 1) cùng với dữ liệu dùng để biểu diễn nó. Hãy kiểm tra lại giá trị ở trong mỗi trường. (Chúng tôi sẽ trình bày cách tìm vector pháp tuyến ở phần sau).



HÌNH 6.14 Một hình tứ diện cùng với dữ liệu biểu diễn nó

Công việc đầu tiên là xây dựng một giải thuật để vẽ đối tượng lưới đa giác này. Đương nhiên bài toán sẽ quy về vẽ từng mặt của lưới đa giác. Phần hiện thực của phương thức `Mesh::draw()` phải duyệt qua mảng các mặt của lưới đa giác. Với mỗi mặt, phương thức trên gửi danh sách các đỉnh và các vector pháp tuyến xuống đường ống đồ họa. Trong OpenGL, chúng ta chỉ định vector pháp tuyến **m** cho đỉnh bằng câu lệnh `glNormal3f(m.x, m.y, m.z)`. Mã giả cho phương thức `Mesh::draw()` như sau:

```
for (each face f in the mesh)
{
      glBegin(GL_POLYGON);
      for(each vertex v in face f)
      {
            glNormal3f(normal at vertex v);
            glVertex3f(position of vertex v);
      }
      glEnd();
}
```

Hình 6.15 là phần hiện thực chi tiết của giải thuật trên.

```
void Mesh::draw()
{
     for (int f = 0; f < numFaces; f++)
     {
           glBegin(GL_POLYGON);
           for (int v = 0; v < face[f].nVerts; v++)
           {
                 int    in = face[f].vert[v].normIndex;
                 int    iv = face[f].vert[v].vertIndex;
                 glNormal3f(norm[in].x, norm[in].y, norm[in].z);
                 glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
           }
           glEnd() ;
```

```
        }
}
```

HÌNH 6.15 Dùng OpenGL hiện thực phương thức vẽ lưới đa giác

Chúng ta cũng cần phương thức tạo lưới đa giác hoặc đọc lưới đa giác được định nghĩa trong một tập tin vào trong bộ nhớ. Trong phần tiếp theo, chúng tôi sẽ giới thiệu một số họ các đối tượng thú vị có thể dùng lưới đa giác để biểu diễn, đồng thời xem xét cách tạo ra chúng. (Đọc và viết tập tin sẽ được trình bày trong bài thực hành 6.1)

Dùng SDL để tạo và vẽ đối tượng lưới đa giác

Trong các ứng dụng đồ họa, để thuận tiện, người ta thường tạo lưới đa giác từ dữ liệu được mô tả bằng ngôn ngữ SDL trong một tập tin. (Trong chương 5, chúng tôi đã trình bày về SDL). Để làm được điều này, chúng ta chỉ cần dẫn xuất lớp `Mesh` từ lớp `Shape` và thêm phương thức `drawOpenGL()`. Chúng ta định nghĩa lại lớp `Mesh` trong hình 6.13 như sau:

```
class Mesh : public Shape
{
      // giống như trong hình 6.13
      virtual void drawOpenGL()
      {
            tellMaterialsGL();
            glPushMatrix();
            glMulMatrixf(transf.m);
            draw();
            glPopMatrix();
      }
};
```

Chúng ta sửa lớp `Scene` để nó chấp nhận từ khóa `mesh` khi đọc tập tin SDL. Theo sau từ khóa này là tên tập tin. Ví dụ để tạo và vẽ phiên bản tịnh tiến và co dãn của lưới đa giác mô phỏng một con tốt được chứa trong tập tin `pawn.3vn`, chúng ta sử dụng:

```
push   translate 3 5 4   scale 3 3 3   mesh pawn.3vn   pop
```

6.3 KHỐI ĐA DIỆN

Chúng ta thường đưa ra những ràng buộc đối với lưới đa giác để cho nó biểu diễn **khối đa diện** (polyhedron). Rất nhiều đối tượng khối rắn mà chúng ta quan tâm thực ra đều là khối đa diện và các giải thuật xử lý lưới đa giác sẽ đơn giản đi rất nhiều nếu chúng chỉ biểu diễn khối đa diện.

Định nghĩa về khối đa diện có thể sẽ hơi khác nhau một chút trong những ngữ cảnh khác nhau, nhưng chúng ta sẽ sử dụng định nghĩa sau

ĐỊNH NGHĨA : **khối đa diện** (polyhedron) là lưới liên thông của các đa giác phẳng đơn giản. Các đa giác này bao bọc một không gian giới hạn.