3次元仮想空間への直接操作

Direct manipulation for 3D virtual space

大芝 崇 Takashi OSHIBA
光延 秀樹 Hideki MITSUNOBU
田中 二郎 Jiro TANAKA

筑波大学
University of Tsukuba

概 要

3次元仮想空間におけるオブジェクトに対して何らかの操作を行う場合、アイコンやメニュー等を用いた操作は、ユーザにとってその操作性が悪いという問題がある。本研究では、3次元仮想空間上のオブジェクトの例として、ルービックキューブを取り上げた。既存のルービックキューブ操作システムの多くはオブジェクトを2次元的に表現したものであった。3次元的に表現したシステムもあるが、その操作は直観的とは言えず、分かりにくいものであった。そこで本研究では、オブジェクトを3次元仮想空間上に表現し、直接操作により扱う手法を提案する。アイコンやメニュー等は一切用いず、マウスのみの操作で結果が即座に視覚化されるシステムを OpenGL を用いて実際に作成した。

1 はじめに

近年の3次元グラフィックスは、ゲーム、映画からマルチメディア、デザイン・工業分野その他で、かつてない注目を浴びており、非常に高機能なグラフィックスライブラリ等も開発されている。しかし、そういったソフトウェアを用いて構築されたシステムの中には、システムの操作法を理解することが難解であるものが多い [1,2]。例えば、3次元仮想空間におけるオブジェクトに対して何らかの操作を行う場合を考えてみると、アイコンやメニュー等を用いた間接的な操作体系では、システムの操作法も難解である。

本研究では、3次元仮想空間上のオブジェクトの題材として、ルービックキューブを取り上げた [3]。その理由は、既存のルービックキューブ操作システムのほとんど [7,8,9,10,11,12,13,14] は、操作性が直観的とは言えず、分かりにくいものであったからである。ルービックキューブのような多関節を持ったオブジェクトへの操作には、「回転」の種類がたくさんあるために操作法が複雑になり直観的でないと言う問題がある。

そこで本研究では、オブジェクトを3次元仮想空間上に表現し、このオブジェクトを直接操作 [4] により扱う手法を提案し、その手法を実装した直観的なシステムを作成した。ルービックキューブへの操作には、3次元仮想空間におけるオブジェクトへの重要な操作の1つである「回転」と言う操作が色々なバリエーションを持って含まれているため、本研究で作成したシステムの操作性を考察することは、3次元仮想空間上の一般的なオブジェクトへの直接操作に対する操作性を考察するための有用な手段となることが考えられる。

2 既存のルービックキューブ操作システムの問題点

既に、”Virtual Rubik's Cube” などと銘打ったルービックキューブ操作システムは Web 上やシェアウェアとして数多く世の中に出廻っており [7,8,9,10,11,12,13,14]、オブジェクトを2次元的に表現したもの [8,10,13] とオブジェクトを3次元的に表現し

たもの [7,9,11,12,14] がある。

オブジェクトを２次元的に表現したものは、「単に菱形が並んでいる」といった感があり、自分が今どこのキューブを触っているのかを把握することは難しい。

２次元であるために、今見えている面の裏側の状況の把握が非常に難しい。また、回転に伴う各キューブの動きがアニメーションによって描画されない（再描画が瞬時に行われる）ものもあり [8,11,13]、これはユーザへのフィードバックに問題がある。

オブジェクトを３次元的に表現したものにも色々と問題点が挙げられる。まず、射影変換に正射影を用いているものがある [7,11] が、この場合時々キューブの表裏が逆転してしまったような錯覚に陥ることがある。射影変換には透視射影を用いるべきである。

また、これはオブジェクトを２次元的に表現したものにも言えることであるが、回転などのオブジェクトへの操作をアイコンやメニュー、キーボードなどを用いて行うものが多く [7,8,9,10,11,12,13,14]、こういった間接的な操作体系は、物理的な操作というよりもむしろ構文的な操作であり [6]、我々が現実世界で実際にルービックキューブを手に持っているような感覚（リアリティ）が希薄になってしまうためユーザにとってその理解が難解になってしまっている。

オブジェクトを３次元的に表現したものには、キューブの回転などの操作をマウスを用いて直接行えるもの [7,9,12,14] も多いが、ここにもユーザを困惑させる原因が潜んでいる。まず、ルービックキューブ全体を回転させる操作（今見えていない面を見るために行う）と、９個のキューブだけをルービックキューブ全体に対して相対的に回転させる操作（各面の色を合わせるため）の区別が分かりにくい。多くのシステムはアイコンを用いて対処しているが、中にはセンターキューブ（各面の真ん中にある１色しか持たない面）をつまんで動かした時は全体の回転で、その他のキューブをつまんで動かした時には９個のキューブの相対的な回転が行われるというもの [12] もあり、この仕組みは現実世界の適切な比喩になっておらず、ユーザを混乱させてしまう。

3 本研究の方針

上で挙げたような問題点を考慮に入れ、本研究では以下のような方針でシステムの実装を行うことにした。

- オブジェクトは３次元的に表現する。
- 射影変換には透視射影を用いる。
- オブジェクトへの操作には、アイコンなどの間接的な操作体系は用いず、マウスのみを用いることにより、オブジェクトを直接操作する。
- ルービックキューブ全体を回転させる操作と、９個のキューブだけを相対的に回転させる操作とは、それぞれを異なるマウスボタンに割り当てることにより区別する。
- 回転の操作にアニメーションを用いることにより、フィードバックを向上させる。

以上は既存のシステムの問題点を改善させるためのものであるが、以下のような追加機能も実装し、さらなる操作性の向上 [5] をはかった。

- 現在触っている面の黒枠の色を少し白くすることにより、フィードバックを向上させる。
- 現在触っている面に、可能な回転方向を示す矢印を取付けることにより、ユーザが混乱することを避ける。
- ９個のキューブだけの相対的な回転の操作に関して、1/4 回転 (±90°) だけでなく、2/4 回転 (半回転。±180°) や 3/4 回転 (±270°)、4/4 回転 (１回転。±360°) などの回転も出来るようにした（ほぼ総てのシステムでは 1/4 回転しか許していない）。
- もし相対的な回転の途中でマウスボタンをリリースしても、リリースされた時点での適切な位置に９個のキューブを揃えてくれるようにする。

4 作成したルービックキューブ操作システム ‘Cubism’

本研究で作成したルービックキューブ操作システム ‘Cubism’ について述べる。‘Cubism’ は汎用３次元グラフィックスライブラリ OpenGL を用いて作成されており、現在 Unix (IRIX 6.3, Solaris 2.5.1), Windows95 上において動作する。

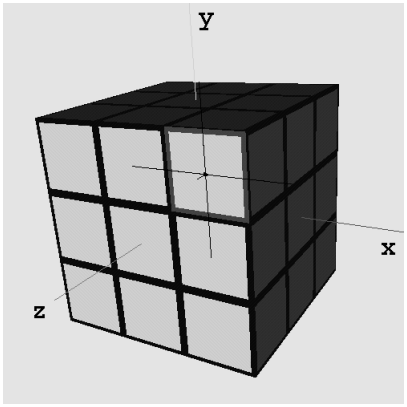

図 1：触った面の黒枠の色が変化。
可能な回転方向を示す十字の印が付く。

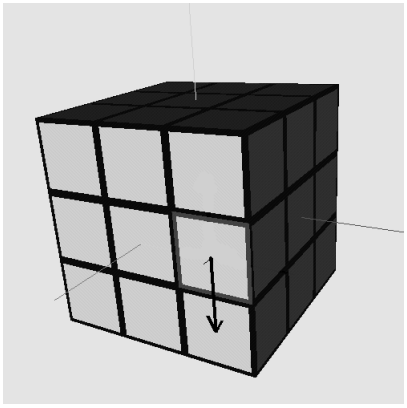

図 2：隣の面を触ると回転方向が決定し、その方向に矢印が出る。

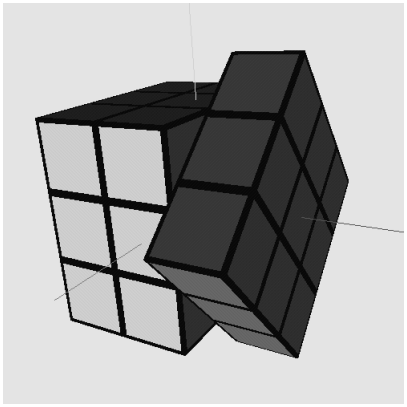

図 3：9 個のキューブの相対的な回転をしているところ。

現段階で実装されている機能は、キューブ全体の回転と、触った面の黒枠の色を変化させることと、その面に矢印を付けること、それと 9 個のキューブの相対的な回転である（図 1,2,3。ルービックキューブは常に 9 個のキューブが一度に回転する）。回転に関する操作は、すべてマウスにより行う直接操作である。

5 直接操作インタフェース

本システムの操作はアイコンやメニュー等は一切用いず、すべてマウスのみの操作で行い、結果は即座にアニメーションにより視覚化される。

5.1 操作法

全体の回転について 適当なキューブ（どれでも良い）にマウスカーソルを合わせ、マウスのミドルボタンのクリック & ドラッグによってルービックキューブ全体の回転を行う。

相対的な回転について

a. 回転させるキューブの選択 マウスカーソルを回転させるキューブの 1 面に合わせると、選択された面の外側の、黒枠の部分の色が少し白っぽくなり、ユーザにキューブの選択が行われたことがフィードバックされる。また、選択された面に十字の印が付き、可能な回転方向が示される（図 1）。

b. 回転方向と回転させるキューブの列の指定 現在マウスで触っているキューブから、回転させたい方向にある隣接したキューブを触ることにより、自動的に回転方向が選択され、同時に回転が施される 9 個のキューブが決定する。それと連動して可能な回転方向が矢印によってユーザにフィードバックされる（図 2）。

c. 相対的な回転の実行 マウスのレフトボタンをクリック & ドラッグし、マウスカーソルを回転方向に動かすことにより、マウスをクリックした位置からの距離（ピクセル数）が回転角度に変換されて、選択された 9 個のキューブの相対的な回転を行う（図 3）。1/4 回転以上の回転（例えば半回転）も可能なので、いちいち 1/4 回転を何度も繰り返さなくても 1 回でユーザの要求する操作が出来る。もし回転の途中でレフトボタンをリリースしても、リリースされた時点での適切な位置に 9 個のキューブを揃えてくれる。

5.2 原理, 実装方法

全体の回転 ルービックキューブの重心は常に 3 次元仮想空間内の原点に位置し、図 1 のように原点から x, y, z 軸（基底ベクトル）が出ている。

本システムでは、マウスの座標の変位 (m_x, m_y) を回転角度に利用し、基底ベクトル自体を回転させている (式 (1)) ので、その結果基底ベクトルはルービックキューブの現在の姿勢を示すことになる。

$$\boldsymbol{b}' = \boldsymbol{R}_y(m_x) \cdot \boldsymbol{R}_x(m_y) \cdot \boldsymbol{b} \qquad (1)$$

$$\boldsymbol{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta\pi/180) & -\sin(\theta\pi/180) \\ 0 & \sin(\theta\pi/180) & \cos(\theta\pi/180) \end{bmatrix}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos(\theta\pi/180) & 0 & \sin(\theta\pi/180) \\ 0 & 1 & 0 \\ -\sin(\theta\pi/180) & 0 & \cos(\theta\pi/180) \end{bmatrix}$$

（ここで、m_x, m_y: マウスカーソルが移動したピクセル数, $\boldsymbol{b}'$: 回転後の基底ベクトル, $\boldsymbol{b}$: 回転の直前の基底ベクトル）

相対的な回転 すべての面に、一意な番号（面が含まれているキューブの座標値と面の色をコード化したもの）を付けておく。今、あるキューブの 1 つの面を触っているとして、次にその隣接する面を触ったとすると、さっき触っていた面と現在触っている面の番号を記憶しておけば、番号の変位から回転軸と回転するべき 9 個のキューブを算出することができる。

図 3 のように、選択されたキューブを y 軸の周りに回転する時は、全体の回転の式 (式 (1)) の $\boldsymbol{R}_y(\theta)$ を用いた式 (式 (2)) を用いる。

$$\boldsymbol{c}' = \boldsymbol{R}_y\left(\sqrt{{m_x}^2 + {m_y}^2}\right) \cdot \boldsymbol{c} \qquad (2)$$

（ここで、$\boldsymbol{c}'$: 回転後のキューブの座標, $\boldsymbol{c}$: 回転の直前のキューブの座標）

マウスをクリックした位置からの距離（ピクセル数）が回転角度になるので、マウスの移動距離を多くすればするほど回転角度も増加し、1/4 回転以上の回転（例えば半回転）も可能になる。

ただし、全体の回転では基底ベクトル自体が回転したが、式 (2) による回転はキューブの座標値が変化するだけで、基底ベクトルは変化しない。

6 まとめ

本研究では、既存のルービックキューブ操作システムの問題点を指摘し、その解決策として 3 次元仮想空間上のオブジェクトをマウスのみを用いた直接操作により扱う手法を提案し、実際にその手法を実装したシステムを作成した。

7 参考文献

[1] Debbie Myers: IRIS Showcase User's Guide 日本語版, 日本シリコングラフィックス株式会社.

[2] James M. Hebert: LIGHTWAVE 3D USER GUIDE, NewTek 社.

[3] P. ジョンソン: ケーススタディ 1. ルービックキューブ, 対話システム設計に適用されるタスク分析, ヒューマンインタフェースの設計方法, pp.201-216.

[4] 光延 秀樹, 岡田 潤, 田中 二郎: 三次元インタラクティブ編集環境の構築, 並列・分散処理研究推進機構 成果概要 (1997), pp.130-131.

[5] D.A. ノーマン: 可視性とフィードバック, 誰のためのデザイン？, 新曜社, pp.158-168.

[6] Ben Shneiderman: 直接操作の原理, ユーザインタフェースの設計方法, 日経 BP 社, pp.126-161.

[7] Glen Nakamura: Glen Nakamura has several cube-like java puzzles, http://www2.hawaii.edu/~gnakamur/puzzle/frame.html

[8] Michael Schubart: Michael Schubart's java cube, http://www.best.com/~schubart/rc/

[9] Junhao Xiong: Junhao Xiong's java cube, http://ucunix.san.uc.edu/~xiongjo/mytest2.html

[10] Karl Hornell: Karl Hornell's java cube, http://www.tdb.uu.se/~karl/java/rubik.html

[11] Geert-Jan van Opdorp: Geert-Jan van Opdorp's java cube, http://www.aie.nl/~geert/java/public/Rubik.html

[12] Song Li: Song Li's java cube, http://www.cs.umbc.edu/~sli2/cube/cube.html

[13] Justin Campbell: Justin Campbell's java cube, http://www.reed.edu/~jmc/project

[14] Kevin Leeds: Kevin Leeds' interactive java cube, http://www.math.gatech.edu/~leeds/java/MoCube/MoCube.html