2003年度　卒　業　論　文

# Shockwave3Dにおける影の生成

指導教員：渡辺　大地

メディア学部 Web3Dプログラミングプロジェクト
学籍番号　00P234
高草木　隼

**2003 年度　　卒　業　論　文　概　要**

**論文題目**

# Shockwave3D における影の生成


| | | | | |
|---|---|---|---|---|
| **メディア学部**<br>**学籍番号：00P234** | **氏名** | 高草木　隼 | **指導教員** | 渡辺　大地 |



| | |
|---|---|
| **キーワード** | Shockwave3D、影、Web3D 技術、Web コンテンツ |



3D コンピュータグラフィクスが映画やテレビゲーム等で頻繁に使用されていることは周知の事実だが、近年ではインターネット上においても 3D コンピュータグラフィクスが扱われ、それらを含むコンテンツが増えてきている。このインターネット上で 3D コンピュータグラフィクスを扱う技術は一般に Web3D 技術と呼ばれ、現在では様々な特徴を持った Web3D 技術が複数開発されている。これによって 3D コンテンツの制作が今後ますます容易になることが予想できるが、この Web3D 技術では 3D コンピュータグラフィクスの研究において重要な課題である影の表現があまり実現されていない。影の表現は空間内の物体の位置関係の把握や 3D シーンへのリアリティの付与において重要であると言われており、影の生成に関する研究は古くから成されているためこれまでに多くの影の生成方法が発表されている。そこで本研究では、Web3D 技術の中でも特に「Shockwave3D」を採り上げ、Shockwave3D において影を生成する手法を提案する。本研究では Shockwave3D の機能を利用して影を生成する既存の手法を拡張した。この手法は影を落とすオブジェクトとまったく同じ形状をしたコピーを作り、それを押し潰して平面上にしたものを影として表現する。本研究ではここからさらに光の入射角度を自由に変更できるようにし、それに伴う影の位置や形状の変化を実現した。また、考慮されていなかった光が差す方向の動的変更を可能にした。これらを踏まえた上で、Shockwave3D による Web コンテンツ上でリアルタイムに影を生成することを実現した。

# 目 次

# 第 1 章

# はじめに

近年では 3D コンピュータグラフィクスを扱ったインタラクティブなコンテンツを含む Web サイトが増えてきている。この背景には 3D コンピュータグラフィクスの技術の進歩が挙げられ、特にインターネット上におけるこの技術は一般に Web3D 技術と呼ばれる。また横川 [1] は、様々な特徴を持った Web3D 技術が複数開発されることでそれぞれの進歩を促し、これを利用した 3D コンテンツが表現手段の一つとして用いられるようになっていると述べている。

その技術の中の一つに、Macromedia 社によって開発された「Shockwave3D」がある。これを利用して作られた 3D コンテンツは動作がとても軽快で、さらに Lingo という独自のプログラミング言語によってインタラクティブ性を持たせられることができる。その代表的な例として、Shockwave3D によって作られたオンラインゲームを集めた Web サイト [2] がある。Shockwave3D は 3D 空間を容易に扱うことができ、3D モデルやカメラ等の基本的な制御には専用のスクリプトが予め用意されている。光が当たった物体の明暗も自動的に表現され、それはユーザ側のコンピュータでリアルタイムにレンダリングされる。しかし現状の Shockwave3D は 3D コンピュータグラフィクスの研究において重要な課題である、影の表現を実現する機能を持っていない。そのため、影の表現を実現するには適切な生成手法が必要である。

影の表現の重要性について Zhang[3] は、影は空間内の物体の位置関係を把握す

る重要な手がかりであり、また 3D シーンにリアリティを与える要素でもあると述べている。影の表現の重要性を例として図 1.1で示す。２枚の画像を見比べると、影に注目することによって球と円錐が空間内のどのような場所に位置しているのか、またどの方向から光が当たっているかということを瞬時に判断することができる。これだけでも十分に影の重要性を証明することができる。このため、影の生成に関する研究は古くからなされており、これまでに様々な生成方法が開発されてきた。最近ではリアルタイム性が重要視され、その技術が映画や 3D ゲーム等で利用されている。

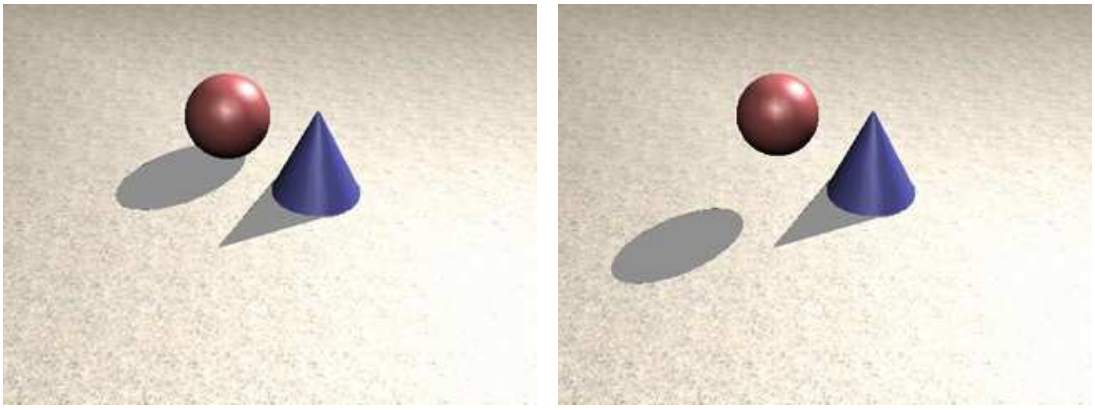

図 1.1: 影が表す物体の位置関係

そこで本研究では、Shockwave3D によって作られた Web コンテンツにおけるリアルタイムな影の生成手法を提案する。3D コンピュータグラフィクスにおける影の生成手法は数多く存在するが、本研究では Swan[4] によって提案された物体のコピーを押し潰す手法を採用した。この手法は Shockwave3D のための手法であり、影を落とす物体のコピーを押し潰すという簡単な処理で影を生成できる。しかし、光の入射角度が一定のまま変更できず、また光が差す方向の動的な変更を考慮していない。本研究ではこれらの問題を解決した上で影が生成できるよう拡張を施した。また、インターネット上で動作するインタラクティブなコンテンツでの利用を考慮するため、擬似的な表現に留める。厳密な正確さは求めないこととし、いかに簡単に且つ高速に影を生成できるかを追及する。

以降、第２章において Shockwave3D に適した影の生成手法を求め、第３章で実際の生成手法を示す。第４章でそれを考察、評価し、最後に展望を述べ締めくくる。

# 第 2 章

# 影の生成手法の探求

## 2.1　既存の生成手法

3D コンピュータグラフィクスの研究において影の生成手法はこれまでに数多く発表されている。本章ではそれらの手法の元になっている基本的な手法の中から複数の候補を挙げて解説し、それらが Shockwave3D に適用できるかを検証する。さらに、Shockwave3D の機能を利用して影を生成する既存の手法を加えて解説し、その実用性を検証する。

## 2.2　影の生成手法の候補

### 2.2.1　レイトレーシング法

レイトレーシング法は Whitted[5] によって提案され、現在では高品質なレンダリング手法として有名であるが、同時に影の生成手法にもなり得る。光線追跡法とも呼ばれ、その名の通りレイ（光線）をトレース（追跡）することで影を生成する。これを図 2.1に示す。手順としてはまず、スクリーン上のピクセルを１つ選び、視点と直線で結ぶ。この直線が視線を表し、この視線と交差する物体があるかどうかを調べる。交差する物体があった場合はその中から最も視点に近いものを選び、次にその交点と光源を線分で結ぶ。この線分が他の物体と交差する場合はその点は影となり、交差しない場合は影ではないことになる。図 2.1において、

点 $A$ は光源との間に他の物体があるので影となるが、点 $B$ は光を遮る物体がないので影にはならない。このようにアルゴリズムは非常に簡単で、さらに各交点において反射、透過、屈折といった光の複雑な現象の表現が可能である。しかし、スクリーンのピクセル数の分だけ計算が必要であり、非常に多くの時間がかかるという欠点がある。品質を上げればそれに比例して計算時間も膨大になる。西田 [6]、鵜飼 [7] は、現在では計算を高速化する方法が多数研究・開発されていると述べている。

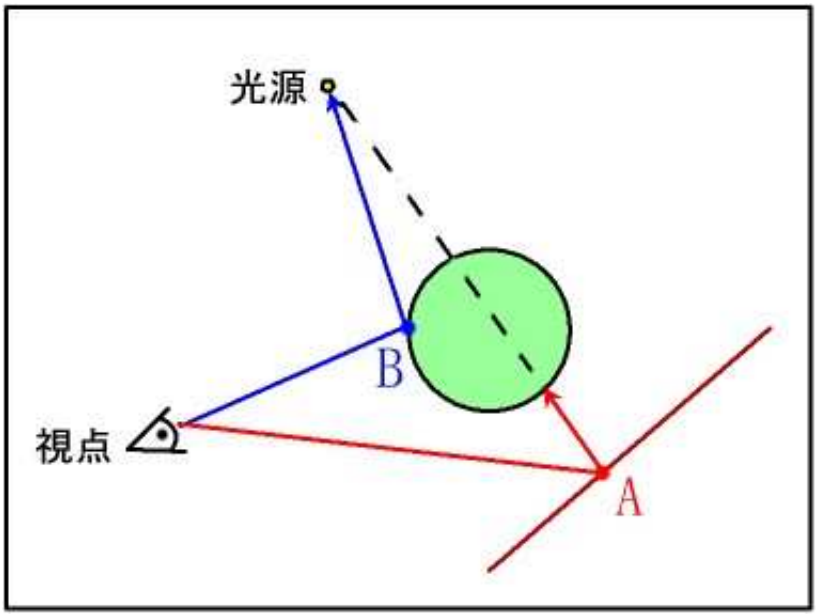


図 2.1: レイトレーシング法

### 2.2.2 シャドウマッピング法

シャドウマッピング法は Williams[8] によって提案された手法である。まず最初に、光源から見たシーンの各点において最も近くにある物体までの距離を求める。これ記録したものを一般にシャドウバッファと呼ぶ。レンダリングする際に、視点から見たシーンの各点の光源からの奥行き値とシャドウバッファに記録された値とを比較する。前者の方が大きい場合、今見ている点よりも光源に近い所に他の

物体があるということなので、その点は影となる。両者がほぼ等しい場合は、その点は光が当たっていて尚且つ視点からも見えるということである。これを図 2.2で示す。この図において、点 $A$ は光源からの奥行き値がシャドウバッファの値よりも大きいので影になり、点 $B$ は二つの値がほぼ等しいので影にはならない。このように、影の生成に必要な計算は二つの値の大きさの比較というとても簡単なものである。しかしこの手法には、光源から見た場合と視点から見た場合の 2 段階の処理が必要である上に、生成する影にエイリアシングが起こるという問題が付随する。特にエイリアシングについては深刻であり、Stamminger ら [9] は広い奥行き範囲を持ったシーンにおいて顕著であると述べている。そのため、この問題を解決するための多くの研究 [9][10] が行われている。また、簡単なアルゴリズム故に Pixar 社の映画で利用されたことがあり、さらには NVIDIA 社の GeForce3 のようなコンシューマレベルのハードウェアに組み入れられている [11]。

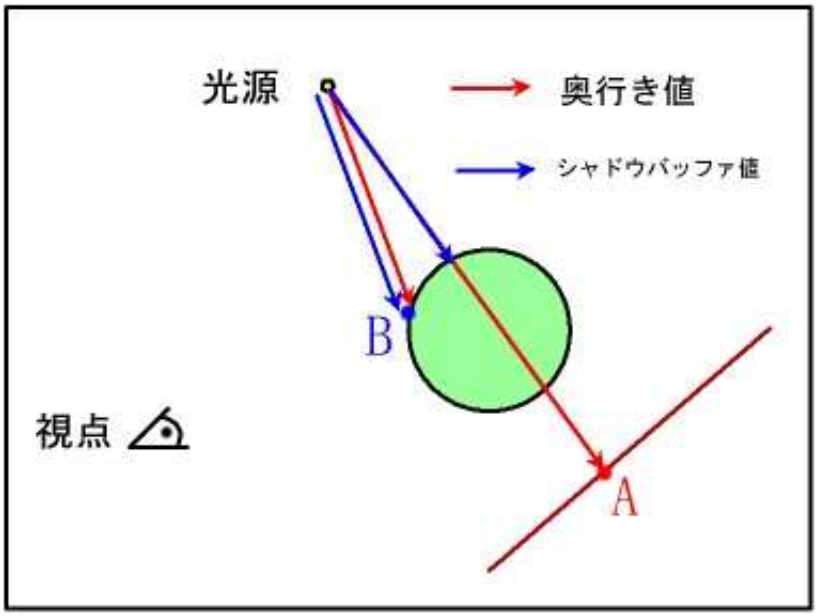


図 2.2: シャドウマッピング法

### 2.2.3　シャドウポリゴン法

シャドウポリゴン法は Crow[12] によって提案された手法である。シャドウボリューム法とも呼ばれるこの手法は、光源と光を遮る物体とで発生する影を生じる空間を利用する。この空間のことをシャドウボリュームと呼び、シャドウボリュームを取り囲む面をシャドウポリゴンと呼ぶ。物体上の影が映る部分は、シャドウボリュームとある面との交差部分として求めることができる。あるいは、物体上のある点がシャドウボリュームの中にあるかどうかで、影になるかどうかを判断できる。これを図 2.3で示す。この手法は、新たに生成するシャドウボリュームの分だけ余計にデータ量を必要とする。しかしながら、近年ではシャドウマッピング法と同じ様にハードウェアに組み入れられることもあり、3D ゲームにおいてはシャドウマッピング法よりもよく使用される傾向にあるという意見もある [13]。また、2003 年の SIGGRAPH においてシャドウマッピング法とシャドウポリゴン法の合成方法が Sen ら [14] によって発表されている。

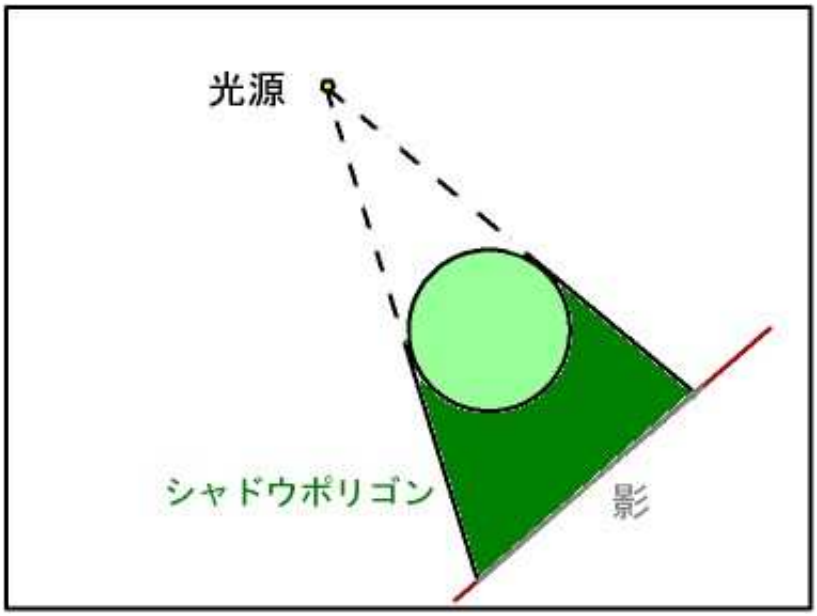


図 2.3: シャドウポリゴン法

### 2.2.4　物体のコピーを押し潰す手法

この手法はSwan[4]によって提案されたShockwave3Dのための影の生成手法である。この手法は影を落とす物体とまったく同じ形状をしたコピーを作り、それを押し潰して平面上にしたものを影として利用する。これを図2.4で示す。影を落とす物体から新たな3D形状を作り出すことがシャドウポリゴン法に似ているが、それを影になるかどうかの判定に利用するのではなくそのまま影として利用してしまう点が異なる。この手法を紹介しているWebサイトのコンテンツ上では、影を落とす物体は常に方向転換をしており、物体のどの方向に光が当たっていてもリアルタイムに影が生成されていることを確認できる。

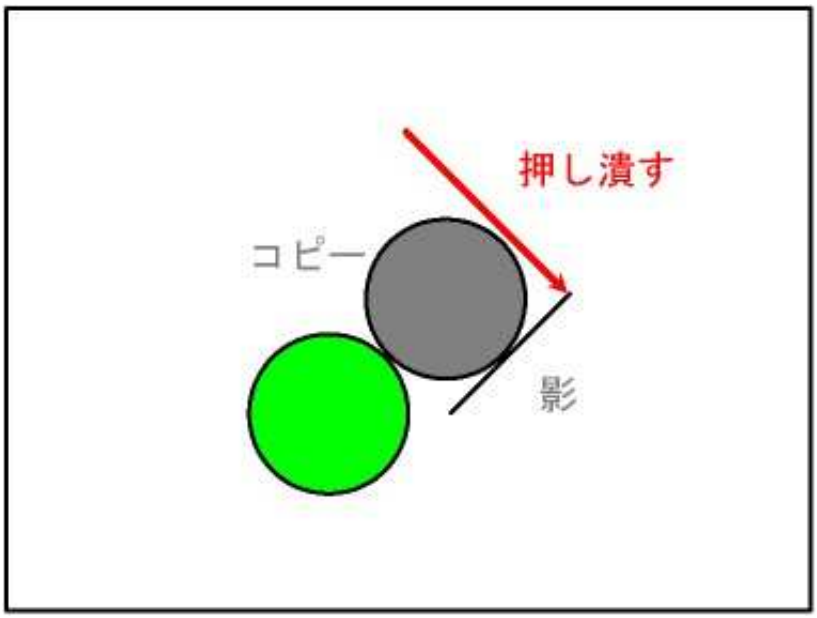


図2.4: 物体のコピーを押し潰す手法

## 2.3　採用した生成手法

前章で述べた通り、本研究では厳密な正確さや見た目の美しさよりも処理の軽さと生成スピードを優先する。それを踏まえた上で、前節の四つの手法のShockwave3Dにおける有用性を検証する。

まず第一にレイトレーシング法だが、光の反射や透過、屈折といった複雑な表現を考えないとすれば非常に簡単なアルゴリズムである。しかし影を落とす物体が少ない場合等で、影にならない部分も含めてスクリーン上の全てのピクセルを処理するため、計算時間が長くなる割に非常に無駄が多くなることもある。これはリアルタイムな生成にとっては大きなデメリットである。

第二にシャドウマッピング法だが、この手法もアルゴリズム自体は非常に簡単だが、問題はシャドウバッファの計算である。既に述べた通り、シャドウマッピング法は主にハードウェア上で使用されており、シャドウバッファもそのハードウェアの持つ機能によって計算される。Shockwave3D にシャドウバッファを計算する機能は無く、それに加えて Web コンテンツで利用できる生成手法が目的なので、コンテンツを閲覧できるユーザの環境を限定してしまうのは好ましくない。

第三にシャドウポリゴン法であるが、上記の二つの手法と違ってピクセル単位の処理を行わないことは有利だが、実際に表示されることの無いシャドウボリュームの生成によって処理が遅くなることが予想される。また、シャドウボリュームと物体の交差部分を求めることが、頂点単位やポリゴン単位の処理をあまり行わない Shockwave3D においては非常に難しいと考えられる。

最後に、物体のコピーを押し潰す手法は元々Shockwave3D のために開発された方法であり、Shockwave3D の持つ機能のみで影を生成し処理自体は影を落とす物体のコピーを押し潰すという非常に簡単なものである。よって、上記の三つの手法に比べてはるかに実用性がある。しかし光の入射角度が一定のまま変更できず、また光が差す方向の動的な変更を考慮していない。

本研究では、生成方法としては物体のコピーを押す潰す手法を採用し、あらゆる Web コンテンツで利用できるようにするためこの方法を拡張することにする。まず光の入射角度を任意の大きさに変更できるようにし、それに伴う影の位置や形状の変化に影の生成を対応させた。さらに光が差す方向の動的な変更を可能にした。Web コンテンツにおいてはインタラクティブ性を考慮し、影を落とす物体の任意方向の方向転換と移動を可能にした。またこの方法は平面に映る影の生成

方法なので、水平な地面に映る影の生成を目的とする。

# 第 3 章

# 影の生成

## 3.1　光源の設定

3D コンピュータグラフィクスで使用される光源には複数の種類があり、それぞれの光源によって発生する影の特徴は異なる。よって、影の生成アルゴリズムは光源の種類に大きく依存することになる。本節では光源と影それぞれの特徴を述べ、本研究において使用する光源を設定する。

光源はまず、それ自体が大きさを持つものと持たないものとに分けられる。大きさを持つ光源の例として線光源や面光源、大きさを持たない光源の例として点光源や平行光線がある。

大きさを持つ光源の場合、発生する影には本影と半影という 2 種類がある。本影とは光源からの直射光がまったく当たらない部分を指し、半影とは光源からの一部の光が当たる部分を指す。逆に大きさを持たない光源の場合は本影のみが発生する。本影は影とそうでない部分の境界線がはっきりしているのに対し半影の境界線はぼやけており、また半影は本影の周囲に発生するので、半影を含む場合は全体的に見ても影の境界線がぼやけている。この違いを図 3.1で示す。本研究では、リアルタイムな生成のため処理の軽減とスピードを重視し、光源を平行光線として扱うこととする。これにより生成するのは本影のみでよいことになる。また、点光源と平行光線はどちらも大きさを持たない光源であるが、図 3.2で示すよ

うに点光源は距離が離れるほど影が広がるという性質がある。平行光線ならばこの表現のための処理をする必要がない。

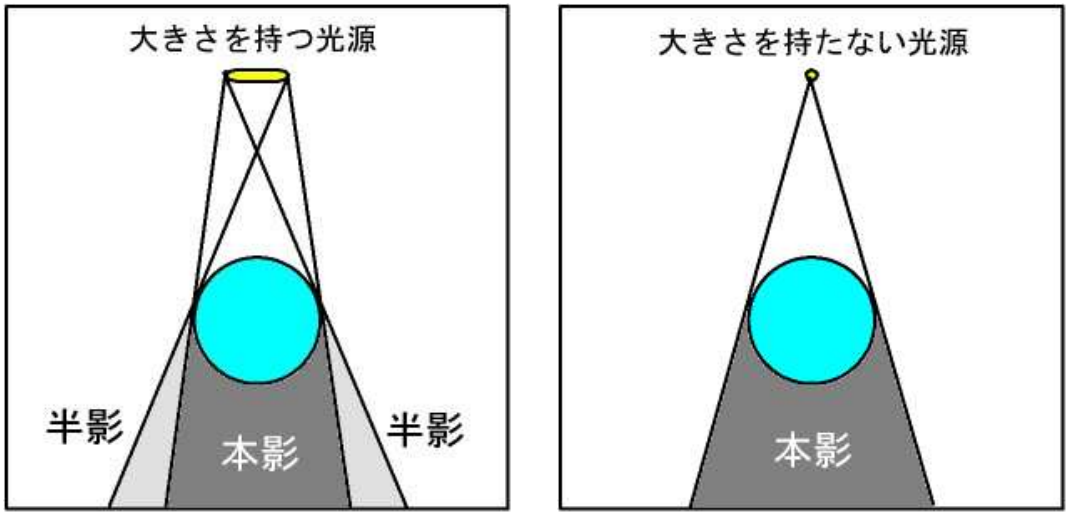


図 3.1: 本影と半影

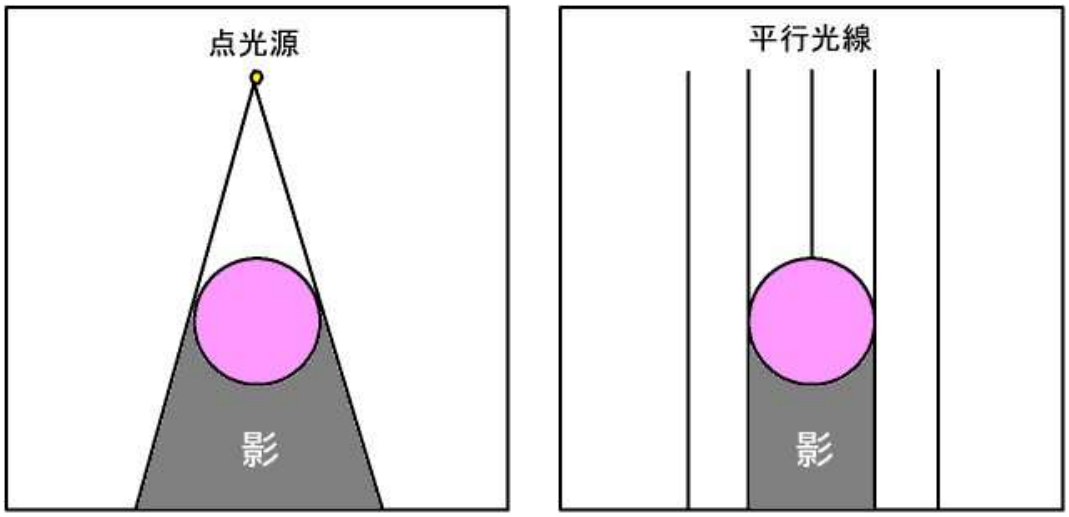


図 3.2: 点光源と平行光線

## 3.2　オブジェクトと空間の定義

本研究において拡張する手法は、光を受けて影を落とす物体とまったく同じ形状をしたコピーを作成し、それを押し潰すことで平面状にして影として表現している。以後、影を落とす物体をターゲットオブジェクト、影として利用するそのコピーをシャドウオブジェクトと呼ぶことにする。本手法では前章で述べた通りこの方法において水平な地面を設定し、図 3.3で示すようにシャドウオブジェクトを作成し、さらに図 3.4のように押し潰して地面上に配置することで影を生成する。その上で光が差す方向や入射角度の変更とターゲットオブジェクトの奥行き方向の移動に対応させ、任意の状況においてリアルタイムな影の生成を実現する。これによって、レイトレーシング法やシャドウマッピング法のようなピクセル単位の処理やシャドウポリゴン法のような表示されないポリゴンの生成をせずにターゲットオブジェクトの形状をそのまま影の形状に利用できる。

また、本研究において定義する空間は Shockwave3D の初期設定に倣い、向かって右・上・手前方向をそれぞれ $x$・$y$・$z$ 軸の正の方向とし、さらに地面を $xz$ 平面とする。

図 3.3: シャドウオブジェクトの作成

図 3.4: シャドウオブジェクトを押し潰す

## 3.3 影の位置と方向

### 3.3.1 オブジェクトの位置座標と光線ベクトル

影を生成するにはまず、シーンの中で影となる部分を求めなければならない。本手法では影となる部分を直接求めるのではなく、予めシャドウオブジェクトという物体として生成するので、光が差す方向や入射角度とターゲットオブジェクトの位置から影が発生する位置を求め、その位置にシャドウオブジェクトを移動させなければならない。

前節において定義した空間において、ターゲットオブジェクトの位置座標を $\mathbf{P}(p_x,\ p_y,\ p_z)$、シャドウオブジェクトの位置座標を $\mathbf{S}(s_x,\ s_y,\ s_z)$ とする。なお、Shockwave3D においてオブジェクトの位置座標とはその中心部の座標を指す。シャドウオブジェクトは押し潰すことで平面状になり地面である $xz$ 平面上に存在するので、$s_y$ の値は 0 に近い任意の値である。これによって影が地面に映る現象を表現できる。以後、影の位置とはシャドウオブジェクトの最終的な位置座標を指すこととする。

また、3.1 節で述べた通り本手法では光源を平行光線として扱う。平行光線には位置という概念が無いので、光線の方向と入射角度を表すベクトル $\mathbf{L} = (l_x,\ l_y,\ l_z)$ を定義する。

### 3.3.2 影の位置

影が発生する位置を求めるにおいて実際に求めるのは $x$ 座標と $z$ 座標だけであるが、計算を簡単にするためそれぞれを別々に求める。まず $x$ 座標を求めるため、点 $\mathbf{P}$、$\mathbf{S}$ とベクトル $\mathbf{L}$ の $xy$ 空間における関係を図 3.5で示す。これらの関係から (3.1) 式が成り立つ。

$$p_y \ = \ \frac{l_y}{l_x}(p_x \ - \ s_x) \tag{3.1}$$

この式は点 $\mathbf{P}$ と点 $\mathbf{S}$ を通る直線を表す式でもあり、これを $s_x$ について解くと (3.2) 式が得られる。

$$s_x \ = \ p_x \ - \ \frac{l_x}{l_y} p_y \tag{3.2}$$

この式によってシャドウオブジェクトの $x$ 座標を求めることができる。また、$z$ 座標についても $zy$ 空間において同様に点 $\mathbf{P}$、$\mathbf{S}$ とベクトル $\mathbf{L}$ の関係を利用すると (3.3) 式が成り立つ。

$$p_y \ = \ \frac{l_y}{l_z}(p_z \ - \ s_z) \tag{3.3}$$

この式を $s_z$ について解くと (3.4) 式が得られる。

$$s_z \ = \ p_z \ - \ \frac{l_z}{l_y} p_y \tag{3.4}$$

(3.2) 式と (3.4) 式によって任意の $\mathbf{P}$ と $\mathbf{L}$ から、つまりターゲットオブジェクトが地面上のどこに位置していても、どのような方向や入射角度で光が当たっていてもシャドウオブジェクトの位置座標を求めることができる。計算に必要な数値は $\mathbf{P}$ の座標と $\mathbf{L}$ の成分だけであり、これらの値自体を求めるのに計算は必要ない。また、計算式自体も非常に簡単な一次関数であるため、リアルタイムな処理に負荷をかけることなく非常に有効である。また、ターゲットオブジェクトの位置と光線の方向、入射角度の関係からシャドウオブジェクトの位置を一意に決定することができる。

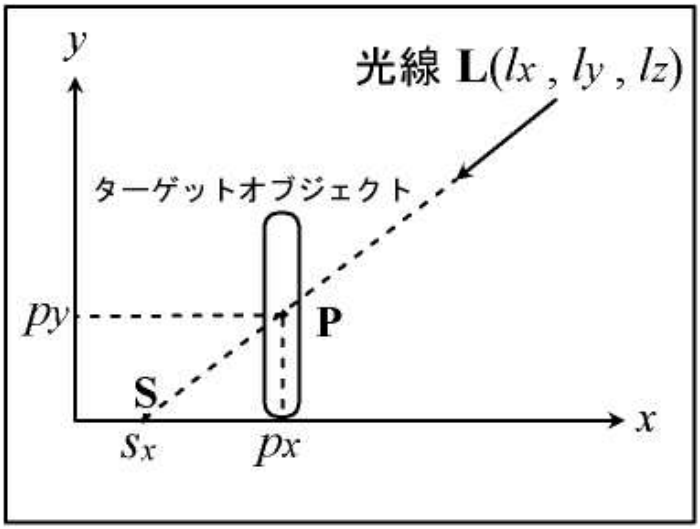


図 3.5: シャドウオブジェクトの $x$ 座標

### 3.3.3 影の方向

影は光が差す方向に向かって伸びる。よって本手法ではシャドウオブジェクトを影が伸びるべき方向へ向けなければならない。この方向とは図 3.6で示す通り $xz$ 平面上におけるベクトル **L** が指す方向である。

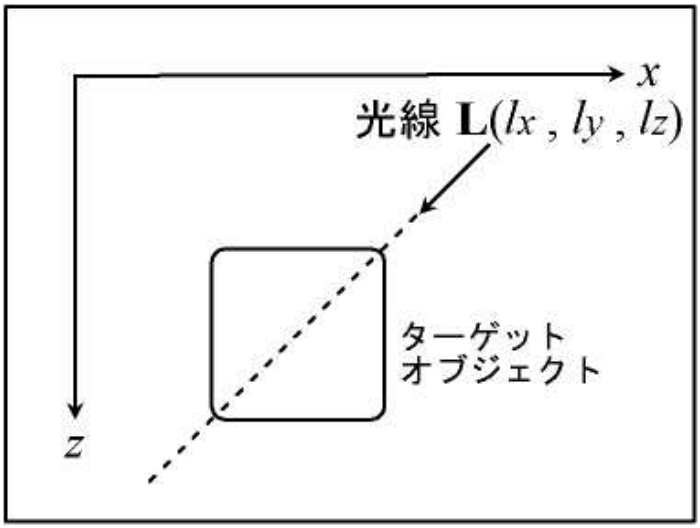


図 3.6: 影が伸びる方向

## 3.4　影の形状と大きさ

### 3.4.1　形状の変化

もしもターゲットオブジェクトが回転体であった場合は、どのような方向や角度から見ても同じ形状に見えるので、光が差す方向や入射角度が変化しても影の形状は変化しない。よって、シャドウオブジェクトを 90 度回転させて地面に倒してから押し潰し、適切な大きさに調整するだけで正確な影の表現を実現することができる。

しかし、一般的にターゲットオブジェクトが回転体である場合は少ない。回転体でない場合はターゲットオブジェクトを見る方向や角度によって見え方は異なる。つまり、光が差す方向と入射角度によって影の形状が変化し、それは光線に沿ってターゲットオブジェクトを見た場合の形状にほぼ等しい。そこで本手法では、回転体であるかどうかに関わらず任意の方向と入射角度から差す光に対して適切な影を生成する。

### 3.4.2　押し潰す方向

本手法ではシャドウオブジェクトを押し潰した時点で影の形状が決定する。よって、平面状にしたシャドウオブジェクトが光線に対して垂直になるように押し潰すことで、光線に沿って見たターゲットオブジェクトの形状をほぼ正確に表現することができる。任意の光線に影の生成が対応するには、光が差す方向と入射角度からシャドウオブジェクトを押し潰す方向を求めなければならない。この方向を図 3.7で示す。この方向はベクトル $\mathbf{L}$ に平行な方向であり、この図はシャドウオブジェクトをこの方向とその逆方向の両方向から押し潰していることを表している。

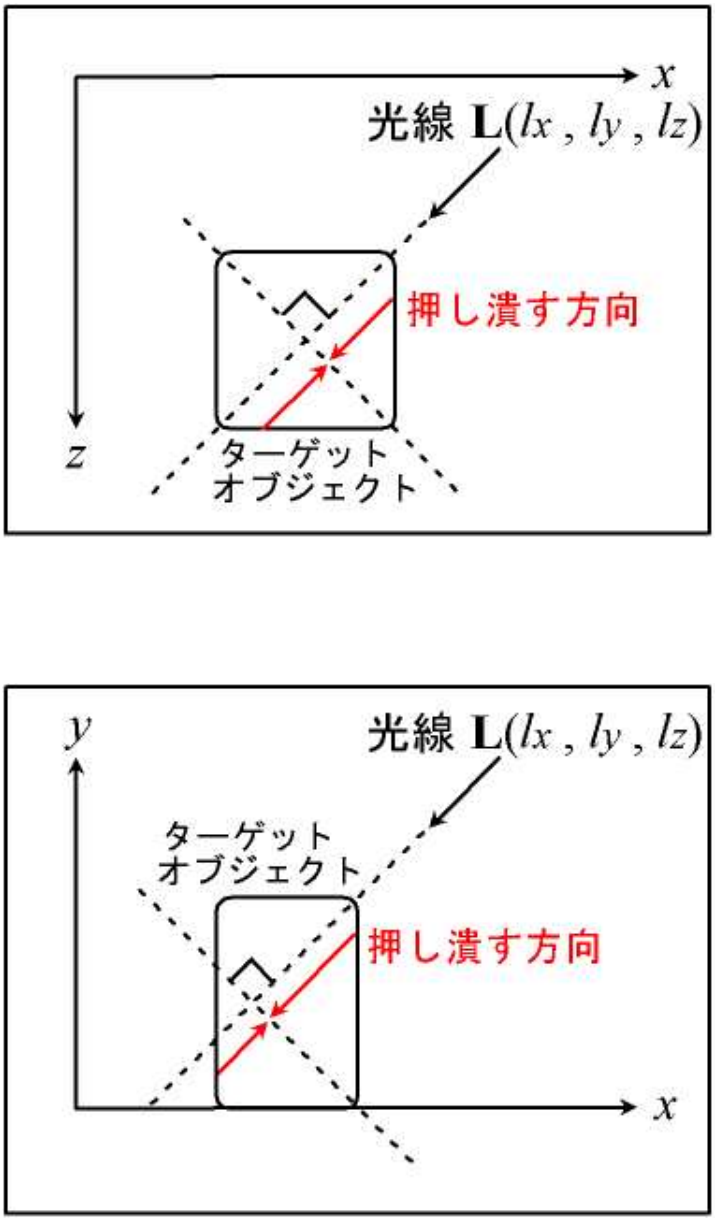


図 3.7: 押し潰す方向

### 3.4.3 大きさの変化

影は光の入射角度によって伸縮する。現実世界においても、太陽の高度が下がる夕方になると影が伸びるという現象を自然に感じることができる。シャドウオブジェクトを押し潰すことで影の形状は決定できるが、これだけではまだ大きさ

の変化が表現できていない。本手法においては光源が平行光線であるため、平面状のシャドウオブジェクトを特定の方向について単純に伸縮させることでこれを表現できる。

### 3.4.4 伸縮率

平面状にしたシャドウオブジェクトを正確な大きさにするには、現状の大きさの伸縮率を求める必要がある。まず $\mathbf{p}'_t$ と $\mathbf{p}'_u$ をそれぞれ平面状のシャドウオブジェクトの上端と下端とし、$\mathbf{s}_t$ と $\mathbf{s}_u$ をそれぞれ正確な影の上端と下端とする。また、光の地面に対する入射角度を $\theta$ とする。これらの $xy$ 空間における関係を図 3.8で示す。ここで求めるべき伸縮率は $\mathbf{s}_t$ と $\mathbf{s}_u$ の距離と $\mathbf{p}'_t$ と $\mathbf{p}'_u$ の距離との比であり、(3.5) 式によって求めることができる。

$$\text{伸縮率} \;=\; \frac{|\mathbf{s}_t - \mathbf{s}_u|}{|\mathbf{p}'_t - \mathbf{p}'_u|} \tag{3.5}$$

また、$|\mathbf{p}'_t - \mathbf{p}'_u|$ と $|\mathbf{s}_t - \mathbf{s}_u|$ の関係を表す (3.6) 式が成り立つ。

$$\frac{|\mathbf{p}'_t - \mathbf{p}'_u|}{|\mathbf{s}_t - \mathbf{s}_u|} \;=\; \sin\theta \tag{3.6}$$

(3.6) 式を (3.5) 式に代入することにより (3.7) 式が得られる。

$$\text{伸縮率} \;=\; \frac{1}{\sin\theta} \tag{3.7}$$

この式からわかるように伸縮率の計算に必要な数値は光の入射角度 $\theta$ だけである。この式によって任意の光の入射角度から伸縮率を求めることができ、シャドウオブジェクトを正確な大きさにすることができる。

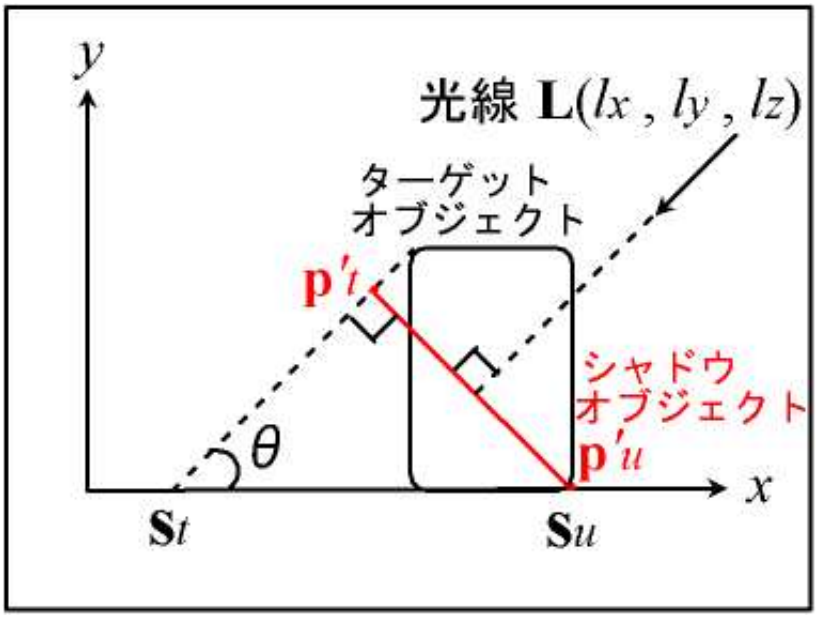


図 3.8: シャドウオブジェクトと正確な影の大きさ

## 3.5 Shockwave3D への実装

### 3.5.1 実装手順

本手法では影を位置、方向、形状、大きさの四つの要素に分けてそれぞれを求めたが、実際の順番は形状、大きさ、方向、位置となる。また、位置については最初に座標を求めるだけであり、シャドウオブジェクトを移動させるのは生成処理の最後においてである。なお、本節において Shockwave3D の命令を下線を付けて示す。

### 3.5.2 実装時における光線ベクトル

Shockwave3D では平行光線を表現できないので、本手法では太陽光と似た性質を持つ directional という光源を擬似的に平行光線として扱っている。この光源は位置座標を持っており、Shockwave3D では全ての位置座標は原点からのベクトルとして扱うことができるので、このベクトルの正反対のベクトルを光線ベクトル

**L** として定義している。

### 3.5.3 実装時における影の形状

3.4 節で述べた通り、本手法では光線に対して垂直な平面になるようにシャドウオブジェクトを押し潰すことで影の形状を決定している。シャドウオブジェクトの生成は clone という命令を用い、押し潰すという処理は scale という命令を用いて特定の方向の大きさを限りなく小さくすることで実現している。scale はオブジェクトの $x$、$y$、$z$ 軸方向それぞれの大きさを設定するという命令であるため、本手法では光線が差す方向が $y$ 軸と平行になるようにシャドウオブジェクトを回転させ、$y$ 軸方向の大きさを限りなく小さくすることで任意の方向に押し潰すことを可能にしている。この方法では光線に対して垂直な平面が地面である $xz$ 平面と平行になるので、光線に沿って見たターゲットオブジェクトの形状を地面を垂直に見下ろした場合のシャドウオブジェクトの形状で再現していることになる。よって平面状になったシャドウオブジェクトの地面への配置が容易になる。

Shockwave3D におけるオブジェクトの回転は $x$、$y$、$z$ 軸のいずれかを回転軸として指定し、rotate という命令を用いて実行する。本手法で必要なシャドウオブジェクトの回転は $y$ 軸と $z$ 軸についてであり、その回転角度は光線ベクトル **L** を利用して求めることができる。図 3.9で示す $\theta_y$ が $y$ 軸について、$\theta_z$ を $90°$ から引いた値が $z$ 軸についての回転角度である。$\theta_y$ は $xz$ 平面に射影した **L** と $x$ 軸によって作られる角の大きさであり、$\theta_z$ は光線の入射角度に等しい。これらの角度の値はベクトルの成す角を計算する angleBetween という命令によって求めることができる。よって $y$ 軸について $\theta_y$、$z$ 軸について $(90° - \theta_z)$ だけシャドウオブジェクトを回転させることで光線が指す方向が $y$ 軸と平行になる。しかし、Shockwave3D において回転したオブジェクトは同様に回転した空間の $x$、$y$、$z$ 軸を持っているため、scale を用いた時にこの回転した軸の方向の大きさが設定されてしまう。この問題は rotate の代わりに preRotate という命令を用いることで解決できる。これは scale 等のオブジェクトの状態を設定する命令よりも先に回転を実行する命令な

ので、回転後のオブジェクトに対して空間の $y$ 軸方向の大きさを設定することができる。よって、この方向の大きさを限りなく小さくすることで任意の光線の方向と入射角度に対応でき、図 3.10のような様々な影を生成できる。

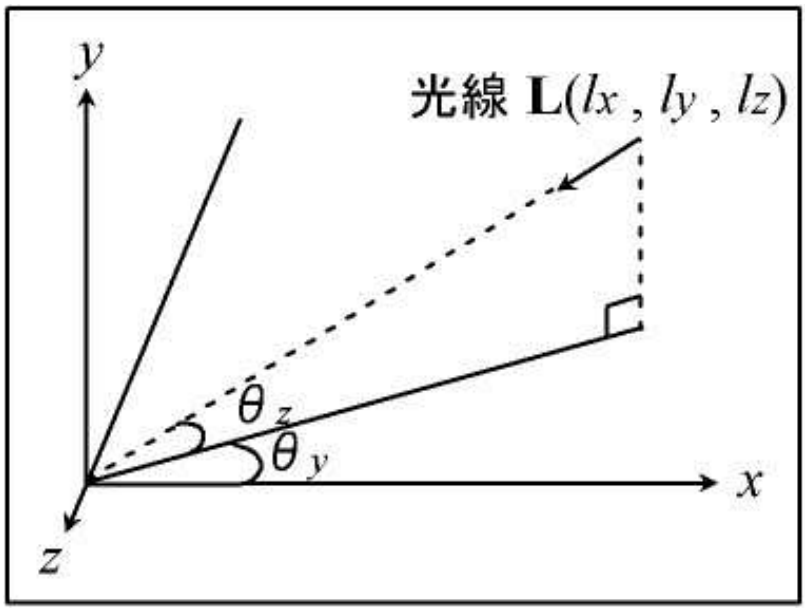


図 3.9: シャドウオブジェクトの回転角度

図 3.10: 任意の光線による影の形状

### 3.5.4　実装時における影の大きさ

影の大きさは 3.4.3 節で述べた通り、特定の方向について伸縮させるだけで決定できる。シャドウオブジェクトはターゲットオブジェクトとまったく同じ位置に作成される。その状態で 3.5.3 節で述べた方法で形状を決定すると、影は $x$ 軸の負の方向に $x$ 軸と平行に伸びている。ここで言う特定の方向とは $x$ 軸の負の方向のことであり、この方向に影を伸縮させる。形状の決定と同様に、オブジェクトを伸縮させたい方向が $x$ 軸と平行になっているので scale を用いることができる。実際は現状の $x$ 軸方向の大きさに 3.4.4 節で求めた伸縮率を掛けた値を設定することで最終的な影の大きさが決定する。

### 3.5.5　実装時における影の方向

影が伸びる方向は 3.3.3 節で述べた通り光が差す方向である。シャドウオブジェクトをこの方向へ向けるには $xz$ 平面上を $y$ 軸について回転させるだけでよい。ここでは回転に rotate を用い、この場合の回転角度は図 3.9で示す $\theta_y$ である。

### 3.5.6　影の生成の完成

scale によってシャドウオブジェクトを押し潰して形状、大きさの順に決定し、その後 rotate で回転させ最後に位置を求めてそこへ移動させることで図 3.11のように影の生成が完成する。影の位置の決定は 3.3 節で述べた通り (3.2) 式と (3.4) 式によって求めることができ、オブジェクトの存在位置を設定する position という命令で実際に移動させている。

図 3.11: 完成した影の生成

### 3.5.7　ターゲットオブジェクトの移動と光線の状態変化

本手法ではターゲットオブジェクトの移動と、光が差す方向や入射角度の変化に影の生成を対応させている。ターゲットオブジェクトは $x$、$y$、$z$ 軸方向に移動させることが可能で、同様に Shockwave3D 上において光源を移動させることで光線ベクトル $\mathbf{L}$ を変化させることができる。ターゲットオブジェクトが $y$ 軸方向に移動した場合は、地面上にある場合の影をターゲットオブジェクトの移動距離だけ光が差す方向に移動させることで対応している。

# 第 4 章

# 考察

## 4.1 本手法の特徴

### 4.1.1 Shockwave3D の持つ機能による影の生成

本研究では、現状では影を表現する機能を持っていない Shockwave3D においてリアルタイムに影を生成する方法を提案した。一般的な 3D コンピュータグラフィクスにおいては高機能なグラフィクスライブラリを用いて影の生成をすることが多いが、それを利用することができない Shockwave3D という限られた環境ゆえに、Shockwave3D の持つ機能以外に何も使用することなく、その機能だけで影の生成を実現した。

### 4.1.2 生成アルゴリズム

影の生成アルゴリズムは第 2 章で紹介した影を落とす物体のコピーを押し潰す手法を拡張し、実用性を高めた。3D コンピュータグラフィクスにおける従来の影の生成方法は、ある部分が影となる条件に当てはまるかどうかを判断するのに対し、本手法は影として表現する物体を先に生成しそれを発生する影の条件に当てはめる。これにより、影だけを処理し影にならない部分はまったく処理をしないので無駄を省くことができた。

## 4.2 評価

### 4.2.1 状況に応じた影の生成

本手法では影を落とす物体と影の二つの座標、光線を表すベクトルによって影を一意に生成することができる。よって影を落とす物体が地面上であるか空中であるか、光がどのような方向や入射角度で当たっているかに関わらず、図 4.1で示すような様々な状況における影を生成できる。

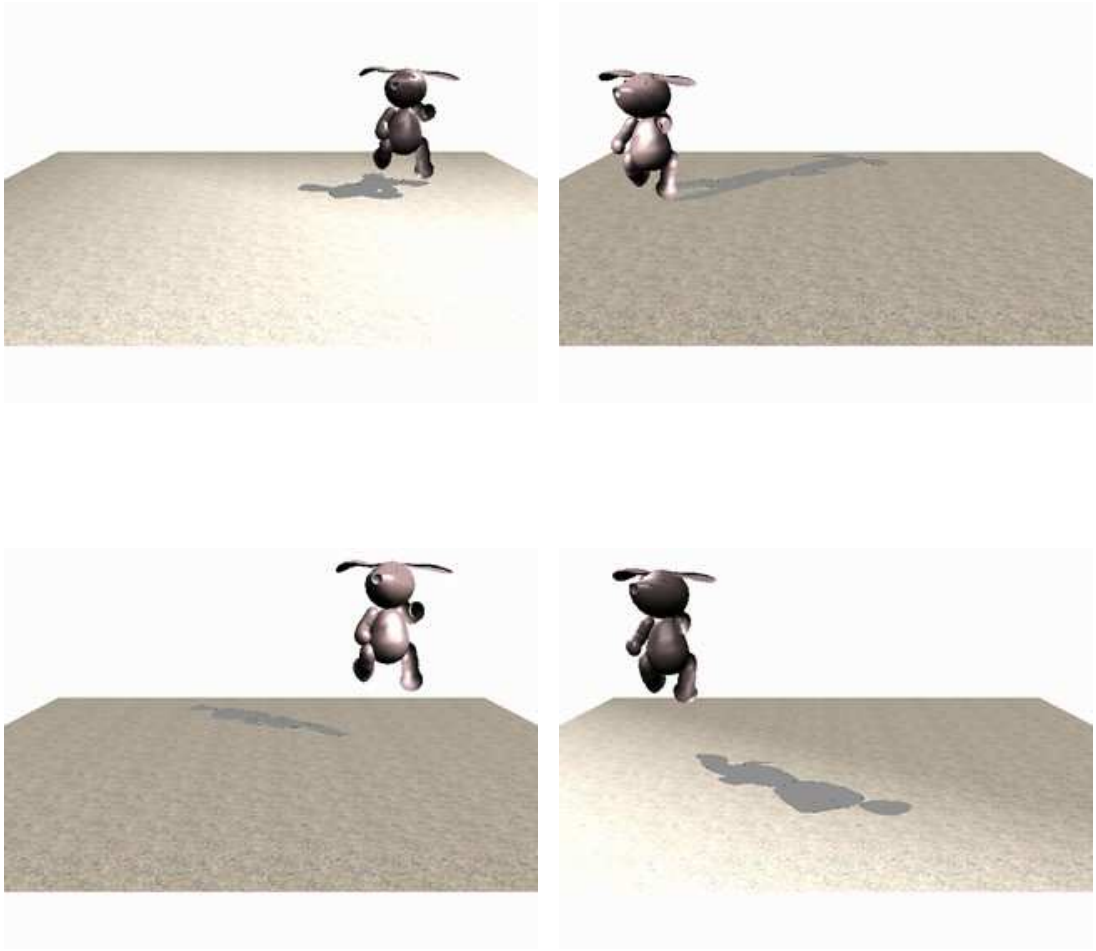

図 4.1: 様々な状況における影

### 4.2.2 Web コンテンツでの利用

インタラクティブな Web コンテンツの制作に特に力を発揮する Shockwave3D の特徴を生かすため、影を落とす物体の 3 次元空間における移動と光線の状態変化を可能にし、それに影の生成をリアルタイムに対応させた。

さらに、本手法を実装した簡単な Web コンテンツを制作し、実際にインターネット上において動作を試みたところ、問題無くリアルタイムな影の生成を確認できた。動作条件については様々な環境における動作実験が必要だが、Shockwave3D 上におけるリアルタイムな影の生成の可能性を示すことができた。

### 4.2.3 複数の物体の影

本手法は複数の物体の影の生成も可能である。これを図 4.2で示す。ある一つの物体の移動は他の物体に何の影響も与えることはなく、また光線の状態変化は全ての物体に一様に影響することが確認できた。物体の数と処理速度の低下の関係や、複数の物体を同時に移動させることについては今後調査が必要である。

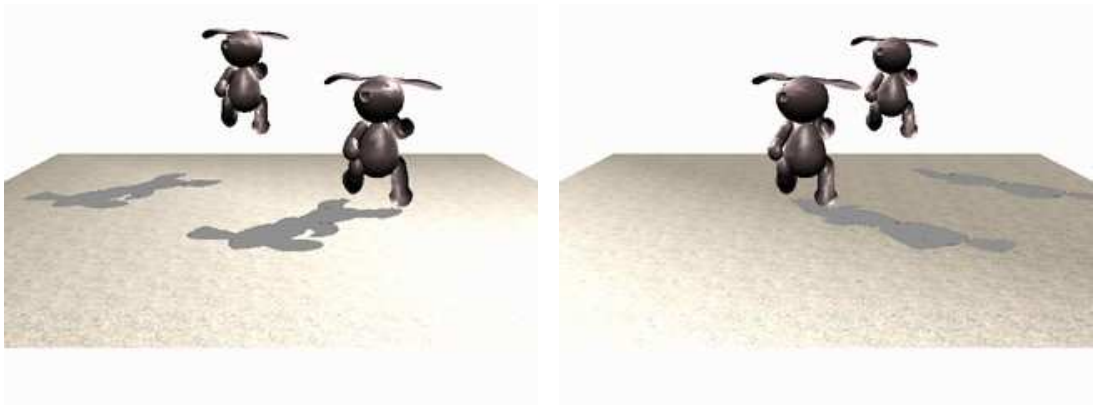

図 4.2: 複数の物体の影

## 4.3 問題点

### 4.3.1 他の物体に映る影

本手法は平面である地面に映る影の生成方法であるが、他の物体に影が映るように拡張することは非常に困難である。そもそも、平面であれば水平でなくても本手法のみで影を映すことはできるが、水平な地面とそうでない地面が混在する場合、それらにまたがって映るような影が生成できない。本手法では影そのものが平面状の物体であるため、平面以外の他の物体の形状に合わせることはとても難しい。

### 4.3.2 光源の種類

本手法では影の生成処理を簡単にするため光源を平行光線として扱ったが、他の種類の光源ではいくつかの問題点が挙げられる。半影の生成については、物体のコピーを複数生成してそれぞれの色の濃さを変え、大きさを変えるか、あるいはずらして重ねることにより擬似的に表現ができると考えられる。しかし境界線がぼやけるという現象の表現がとても困難である。また影の広がりについては、Shockwave3D では特定の軸方向の大きさを全体的に変化させることしかできないので、光源から離れるほど影が大きく広がるようにすることは Shockwave3D の機能では不可能であると考えられる。

## 4.4 展望

インターネット上において 3D コンピュータグラフィクスが扱われることは今後ますます増え、Shockwave3D はこれからもその代表的手段となることが予想される。その Shockwave3D において影の表現を実現することは 3D コンテンツにより一層のリアリティを与えることができ、コンテンツの形態によって他にも様々な恩恵があると考えられる。これはコンテンツの製作者にとってもユーザにとっても非常に意義のあることである。本研究は Shockwave3D においてリアルタイムな

影の生成が可能であることを示唆し、同時に3Dコンテンツの発展の可能性をも示したと言える。

# 謝辞

本研究を進めるにあたり、多大なる助言とご指導を頂いた本校メディア学部の渡辺大地講師、電気通信大学の和田篤氏、3D モデルを提供していただいた本校メディア学部 3DCG アプリケーション構築プロジェクトの斉藤寛明氏に厚く感謝の意を表する。また、本研究において様々なご協力を頂いた本校メディア学部 3DCG アプリケーション構築プロジェクト、Web3D プログラミングプロジェクトの諸氏に厚く感謝の意を表する。

# 参考文献


[1] 横川隆史,“ 各種 3D グラフィックス最適化手法における複数の Web3D での差異に関する研究 ”，東京工科大学メディア学部卒業論文，2003 年 3 月.

[2] shockwave.com, http://jp.shockwave.com/games/3d/3d.html.

[3] H.Zhang,“ Forward Shadow Mapping ”, In Proceedings of the 9th Eurographics Workshop on Rendering 1998, pp.131-138, June 1998.

[4] THE BURROW Director 8.5 Site, http://www.theburrow.co.uk/d85/.

[5] T.Whitted,“ An Improved Illumination Model for Shaded Display ”, Communication of the ACM, Vol.23, No.6, pp.343-349, June 1980.

[6] Nishita's Homepage, http://nis-lab.is.s.u-tokyo.ac.jp/~nis/indxj.shtml.

[7] Ugai laboratory Homepage, http://wsim.cs.ehime-u.ac.jp/index.html.

[8] L.Williams,“ Casting Curved Shadows on Curved Surfaces ”, In Proceedings of ACM SIGGRAPH '78, Vol.12, No.3, pp.270-274, August 1978.

[9] M.Stamminger and G.Drettakis,“ Perspective Shadow Maps ”, In Proceedings of ACM SIGGRAPH 2002, pp.557-562, July 2002.

[10] R.Fernando, S.Fernandez, K.Bala and D.P.Greenberg,“ Adaptive Shadow Maps ”, In Proceedings of ACM SIGGRAPH 2001, pp.387-390, 2001.

[11] Shadow Mapping with Today's OpenGL Hardware,
http://developer.nvidia.com/object/ogl_shadowmap.html.

[12] F.Crow,“ Shadow algorithms for computer graphics ”, In Proceedings of ACM SIGGRAPH '77, Vol.11, No.2, pp.242-248, July 1977.

[13] NVIDIA シェーディング技術の秘密を尋ねる (3),
http://pcweb.mycom.co.jp/news/2003/09/17/21.html.

[14] P.Sen, M.Cammarano and P.Hanrahan, “ Shadow Sillhouette Maps ”, SIGGRAPH 2003, July 2003.