(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-24236
(P2006-24236A)
(43)公開日　平成18年1月26日(2006.1.26)

(51)Int.Cl.		F I			テーマコード（参考）
G06T 15/70	(2006.01)	G06T	15/70	B	2C001
A63F 13/00	(2006.01)	A63F	13/00	C	5B050

審査請求 有　請求項の数 17　ＯＬ　（全 52 頁）

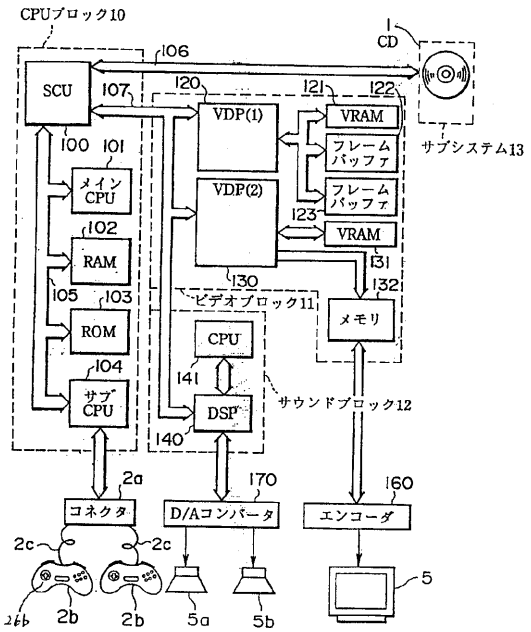
(21)出願番号　特願2005-271433 (P2005-271433)
(22)出願日　平成17年9月20日 (2005.9.20)
(62)分割の表示　特願平10-500414の分割
原出願日　平成9年6月4日 (1997.6.4)
(31)優先権主張番号　特願平8-165224
(32)優先日　平成8年6月5日 (1996.6.5)
(33)優先権主張国　日本国 (JP)
(31)優先権主張番号　特願平8-186666
(32)優先日　平成8年6月27日 (1996.6.27)
(33)優先権主張国　日本国 (JP)

(71)出願人　000132471
株式会社セガ
東京都大田区羽田１丁目２番１２号
(74)代理人　100079108
弁理士　稲葉　良幸
(74)代理人　100080953
弁理士　田中　克郎
(74)代理人　100093861
弁理士　大賀　眞司
(72)発明者　加来　徹也
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　峰　裕一朗
東京都大田区羽田１丁目２番１２号　株式会社セガ内

最終頁に続く

(54)【発明の名称】ゲーム用画像処理

(57)【要約】
【課題】
本発明は、画像処理に要する演算量やデータ量を極力少なく抑えて、よりリアルタイムに、かつ、キャラクタの画面上の動きをよりリアルに表現できるゲーム用画像処理の手法を提供するものである。
【解決手段】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルに仮想的な向心力を与える画像処理手段を備える。
【選択図】　図１



【特許請求の範囲】
【請求項１】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにした画像処理を行うゲーム装置であって、前記モデルに仮想的な向心力を与える画像処理手段を備えることを特徴とするゲーム装置。
【請求項２】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルに仮想的な向心力を与える画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項３】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルが移動している時と静止している時との摩擦量を変えて、このモデルに与える画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項４】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルが置かれるステージの表面形状に合わせて前記モデルの投影像を表示させる画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項５】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記仮想視点が表示対象としての前記モデルに対して成す視野領域と、表示対象として設定されていない他のモデルとの重なり判定を行い、この判定が肯定された場合に、この他のモデルが表示されないようした画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項６】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルの予定された移動軌跡の終了点に対して、この終了点とは異なる目標点が設定され、前記終了点をこの目標点に一致させるように移動軌跡が補間される画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項７】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルが前記仮想空間内に置かれている高低差を求め、この高低差に応じてこのモデルに対して与えられている動作を補正する画像処理手段を備えることを特徴とするゲーム用画像処理装置。
【請求項８】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置であって、モデルの移動軌跡を残像として表現するための残像表現処理手段を備え、この処理手段は前記モデルの現在モーション以前のモーションデータをそのまま記憶する記憶手段と、この記憶データを前記モデルの現在のモーションデータとともに表示する表示制御手段とを備えることを特徴とするゲーム用

画像処理装置。
【請求項９】
請求項８記載の装置において、 前記表示制御手段は、前記記憶手段のデータに演出を加えて表示するための演出手段を備えるゲーム用画像処理装置。
【請求項１０】
請求項９記載の装置において、前記演出手段は、半透明の画像処理を行うゲーム用画像処理装置。
【請求項１１】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置であって、前記モデルの移動特性を算出する算出手段と、前記仮想空間内に飛翔体を存在させるとともに、この飛翔体の一連運動を前記算出手段による算出結果に応じて制御する飛翔体制御手段とを備えることを特徴とするゲーム用画像処理装置。
【請求項１２】
仮想空間内に設定されたモデルの自由落下運動をシミュレートするゲーム用画像処理装置において、前記モデルの全体運動をシミュレートする第１の手段と、前記モデルに周回運動を付与する第２の手段とを備えることを特徴とするゲーム用画像処理装置。
【請求項１３】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置において、不定形の区画を前記仮想空間内に設定する区画設定手段と、この区画とこのモデルとの間に所定のベクトルを設定する手段であり、この区画を越えてこのモデルが移動しようとする場合には、このベクトルをモデルが区画を越えない方向に設定するベクトル設定手段と、このベクトルに応じて前記モデルの移動を制御するモデル移動制御手段と、を備えることを特徴とするゲーム用画像処理装置。
【請求項１４】
所定のパラメータを与えて周期的に変化するテクスチャーを形成する装置において、第１のテクスチャー列を作る第１の手段と、第２のテクスチャー列を作る手段とを備えるとともに、第１のテクスチャー列の最後或いは最初のテクスチャーと、第２のテクスチャー列の最初或いは最後のテクスチャーとが連続する形態になるようにそれぞれの手段にパラメータを与えるとともに、第１のテクスチャー列と第２のテクスチャー列の少なくとも一方の透明度を漸増或いは漸減して両方のテクスチャー列の最初の手テクスチャー同士から順に重ねて目的とするテクスチャー列を形成することを特徴としたテクスチャー生成装置。
【請求項１５】
周期的に変化するテクスチャーを形成する方法において、所定の方向に変化する第１のテクスチャー列を作る第１の工程と、同様に所定方向に変化する第２のテクスチャー列を作る工程とを備えるとともに、第１のテクスチャー列の最後或いは最初のテクスチャーと、第２のテクスチャー列の最初或いは最後のテクスチャーとが連続する形態になるようにするとともに、第１のテクスチャー列と第２のテクスチャー列の少なくとも一方の透明度を漸増或いは漸減して両方のテクスチャー列の最初のテクスチャー同士から順に重ねて目的とするテクスチャー列を形成することを特徴としたテクスチャー生成方法。
【請求項１６】
仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置において、二つのモデル間に所定の条件が成立したか否かを判定する第１の手段と、前記条件が成立したことをそれぞれのモデルの運動に反映させる第２の手段と、この条件が成立したときに一方のモデルの運動の再生速度

を他方のモデルに対して変化させる第３の手段とを備えることを特徴としたゲーム用画像処理装置。

【請求項１７】

仮想空間内に設定されたモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示する画像処理を行うゲーム装置に使用する、前記画像処理のプログラムを記録した記録媒体であって、前記プログラムは前記モデルに仮想的な向心力を与える一連の手順を記載した情報を含むことを特徴とする記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、仮想的に設定したゲーム空間（以下、「仮想空間」と云う。）における画像（以下、「仮想画像」と云う。）の処理に係わり、特に、仮想空間内に設定された仮想なモデル（例えば、キャラクタと称される）の画面上での動きをよりリアルに表現できるゲーム用画像処理に関する。この画像処理の手法は、特に、３Ｄビデオゲーム機に好適である。

【背景技術】

【０００２】

コンピュータ・グラフィックス（ＣＧ）技術の発達により、仮想的に設定した仮想空間（「仮想世界」とも云う。）を立体的にリアルタイムに表現できるようになった。近年開発されている高速演算が可能な中央処理装置(ＣＰＵ)、ビデオ表示プロセッサ(ＶＤＰ)を搭載し、このコンピュータ・グラフィックス技術を高速にかつ経済的に利用しているのがビデオゲーム機の技術分野である。

【０００３】

このビデオゲーム機では、ユーザ（遊戯者或いはプレーヤ、と云っても良い。）の操作内容に基づいてゲームプレイの内容を刻々と変化させるため、仮想空間内において被表示物体を高速で自在に動かす必要がある。このため、通常は、被表示体としてのモデル（例えば、キャラクタ（人物））を三角形或いは四角形のポリゴンと云う多角形の小片の集合から構成し、各ポリゴンの空間位置を同時に変化させながらこれらのモデルの動きを表現している。

【０００４】

また、キャラクタの腕や脚等、被表示物体の特定の部分や面が同一の動きをする場合には、複数のポリゴンを集めたポリゴンデータ群を単位とし、ポリゴンデータ群ごとに空間位置を与え、この特定の部分や面を同時に動かしていた。

【０００５】

近年、キャラクタをポリゴンによって構成し、仮想空間内のキャラクタの動きを所定の仮想視点から捉えた映像をモニタに表現するようにした、いわゆる３Ｄ（３次元）ゲームが市場において注目を集め、特に、複数の闘士の格闘をシュミレートしたものが高い人気を得ている（例えば、株式会社セガ・エンタープライゼス製の「バーチャファイター」（商標））。

【０００６】

この格闘シュミレーションゲームでは、遊戯者はコントローラに付いているスティックやパッド或いは釦を素早く操作して、画面に表示されている闘士が、スティック等の操作によって決められたコマンドに応じて動作するようにされる。闘士の動きは「モーション」と称されており、このモーションを実現するためのデータは、モーションキャプチャー

の技術を用いて取得される。そして、必要に応じてこのデータを加工し、最終的なモーションデータとして３Ｄビデオゲーム機において利用されている。

【発明の開示】

【発明が解決しようとする課題】

【０００７】

この種のビデオゲームにおいては、その商品価値を高めるために、キャラクタの動きをよりリアリティを持って表現することが望まれる。すなわち、例えば、実際の格闘の動作に極力近づけながら、より多彩な動きを付加することである。しかしながら、予想されるキャラクタの動きは極めて多岐にわたるために、このような目的を達成する上で未だ改善されなければならないことが数多くあった。もっとも、望まれるモーションデータを予め全て作成してメモリに記憶させたり、このようなモーションを得るための特性式を得ること等も考えられるが、これではデータ量が膨大になり、リアルタイムの処理が行えない問題がある。

【０００８】

したがって、本発明の主目的は、画像処理に要する演算量やデータ量を極力少なく抑えて、よりリアルタイムに、かつ、キャラクタの画面上の動きをよりリアルに表現できるゲーム用画像処理の手法を提供することである。

【０００９】

この主目的の観点から、従来のビデオゲーム機が有していた問題の具体的な第１の側面は次のようなものである。

【００１０】

３Ｄビデオゲームでは、２次元の画面に仮想視点からの映像が投影変換されて表現されるために、画面の奥行き方向に（仮想空間のｚ方向）、遊戯者が望むように闘士を移動させることは難しく、したがって、遊戯者が一つの闘士を他の闘士の周りを周回させるようにすることは何等配慮されていなかった。そこで、このようなキャラクタの周回の動きを改善することを、本発明の１つの具体的な目的とする。

【００１１】

従来の３Ｄビデオゲーム機は、仮想視点からの映像を表示するようにしていたため、闘士を遮るような位置に、壁等の構造物が仮想空間内に設定されていた場合、キャラクタが遮られて表示されてしまうと云う問題があった。そこで、このようにキャラクタが構造物に遮られるという事態を改善した表示を行うことを、本発明の別の具体的な目的とする。

【００１２】

従来のビデオゲーム機では、キャラクタのモーションを例えばスプライン関数によって順次作って行く方法、又は予め定められたパターンを順次フレームに再現して行く方法が採用されている。しかしながら、従来のビデオゲーム機ではこのモーションが固定されており、相手側キャラクタの動き等に合わせてモーションを補正することができなかった。そこで、このようなモーション補正を行えるようにすることを、本発明のさらに別の具体的な目的とする。

【００１３】

さらに、上記主目的の観点から派生することとして、その商品価値を高めるために、上述した第１の側面に加えて、キャラクタの動きなどに演出効果を高めた画面表示を行うことが望まれている。この要望に鑑みた従来のビデオゲーム機の問題の第２の側面は、具体的には以下のように説明できる。

【００１４】

ＣＧ画像の演出効果を高める技術として、モーション・ブラーが知られている。このモーション・ブラーによれば、一つの画素に多くのレイを発生させ、その平均をとった色を付けることによってピンボケや動きのある絵を作ることができる。 一方、近年では、ビデオゲーム等のＣＧ映像の分野では、さらに演出効果を高めるために、キャラクタの動きに人間の視覚生理上ありがちな残像を一緒に表示することが行われている。例えば、闘士が振り回す刀の軌跡に残像を付すような場合である。そこで、当業者は、残像としてのポ

リゴンを闘士のモーションに合わせて計算して、この残像を闘士の近くに表示するようにしていた。

【００１５】

しかしながら、予想されるキャラクタの動きは極めて多岐にわたるために、全ての場合に合わせて、多くの形態の残像用のポリゴンを作成してこれをメモリに記憶させることは、限られたコンピュータグラフィックス装置の能力を制限することになるばかりか、キャラクタのモーションに合わせて残像のポリゴンを計算することは、コンピュータグラフィックス装置に対する負荷が大きい。そこで、ＣＧ画像を形成する上で、演出効果を高めることの配慮があっても大きな計算負荷を与えることが無く（より詳しくは、このような負荷を少なくして）、残像映像を実像画面と同時に表示することができるようにすることを、本発明のさらに別の具体的な目的とする。

【００１６】

従来の画像処理装置を利用したゲーム装置において、画面に砂や水飛沫等の飛翔体を表示することが行われている（例えば、(株)セガ・エンタープライゼス製「セガラリー」（商標））。しかしながら、このような水飛沫や砂飛沫は、テクスチャーをポリゴンにマッピングしていただけであったため、モデル（車等）の動きを正確に飛翔体に反映させることができなかった。そこで、モデルの動きをより正確に飛翔体に反映させるようにすることを、本発明のさらに別の具体的な目的とする。

【００１７】

従来の画像処理装置では、仮想空間内を落下するモデルの運動をシミュレートする上での品質は十分でなかった。そこで、仮想空間内を自由落下する運動のシミュレートをより高品質なものにすることを、本発明のさらに別の具体的な目的とする。

【００１８】

従来の画像処理装置においては、仮想空間内の区画と移動するモデルとの当たりを判定し、この判定が肯定された場合には、モデルが区画を越えることがないようにその動きを制限していた。通常、この種の区画としては、四角や円形等の定型タイプのものが設定されていた。この場合は、区画が一般的な形状であるために、キャラクタを区画から追い出す運動の計算をすることも容易であった。しかしながら、この区画が不定形の場合には、従来のコンピュータグラフィックス装置では、このことが容易に対応できないという問題があった。そこで、不定形の区画であっても、区画とモデルとの衝突判定肯定後のモデルの動き（モーション）の計算処理が的確且つ容易に実行できるようにすることを、本発明のさらに別の具体的な目的とする。

【００１９】

従来、ゲーム装置に適用される画像処理の分野では、周期的な絵の列（例えば、繰り返す波の絵、縄跳び等）を表現する場合、これらの絵の列を手作業で作成してテクスチャー列とし、これを順次ポリゴンにマッピングすることを繰り返すことにより、同じ動作を繰り返す一連の映像を表現していた。

【００２０】

このようなテクスチャーの列を作成するのは、多くの手作業を要することになるので、アプリケーションソフトウエアを利用することが試みられている。このようなアプリケーションソフトウエアとしては、例えば、商品名：Alias/Wavefr ont（Alias/Wavefront社（110 Richomond St.East Toront,Ontario Canada M5c 1pl）製）が知られている。このソフトウエアを利用する場合、所定のパラメータをソフトウエアに与えれば目的とするテクスチャー列を得ることができる。

【００２１】

しかしながら、従来のこの種のソフトウエアを利用した場合、テクスチャー列の最初と最後、すなわち、テクスチャーの柄（形態）が遊戯者に取って連続しないような絵になることが通例であった。

【００２２】

したがって、従来の画像処理の分野では、繰り返す絵の列を作る上で未だ改良する余地

があった。そこで、画像処理によりこの種のテクスチャー列を作成できるようにすることを、本発明のさらに別の具体的な目的とする。

【００２３】

従来の画像処理装置では、再生画像の演出感を高めるために、キャラクタの動作の再生速度を遅くしてスロー再生を行う場面を有していた。しかしながら、このスロー再生の演出はその効果が限られていた。そこで、より効果的な演出感を醸し出す上での再生速度の変化を可能にしたゲーム用画像処理の手法を提供することを、本発明のさらに別の具体的な目的とする。

【課題を解決するための手段】

【００２４】

本発明によって、その主目的である、画像処理に要する演算量やデータ量を極力少なく抑えて、よりリアルタイムに、かつ、キャラクタの画面上の動きをよりリアルに表現できるゲーム用画像処理の手法が様々な観点から提供される。本発明では、この主目的から派生する種々の具体的な目的を達成するため、以下のような種々の構成が採られる。

【００２５】

本発明の１つの側面を成す一連の構成は以下のようである。すなわち、本発明は、仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、仮想視点に基づくこの仮想空間の映像を表示手段において表示するようにしたゲーム用画像処理装置であって、前記モデルに仮想的な向心力を与える画像処理手段を備えることを特徴とする。また、このゲーム用画像処理装置を備えたゲーム装置が提供される。さらに、この画像処理手段を実行する一連の手順を含む情報を記録した記録媒体が提供される。

【００２６】

さらに、本発明は、この種のゲーム用画像処理装置において、前記モデルが移動している時と静止している時との摩擦量を変えて、このモデルに与える画像処理手段を備えることを特徴とする。

【００２７】

またさらに、本発明は、この種のゲーム用画像処理装置において、前記モデルが置かれるステージの表面形状に合わせて前記モデルの投影像を表示させる画像処理手段を備えることを特徴とする。

【００２８】

さらに発明は、この種のゲーム用画像処理装置において、前記仮想視点が表示対象としての前記モデルに対して成す視野領域と、表示対象として設定されていない他のモデルとの重なり判定を行い、この判定が肯定された場合に、この他のモデルが表示されないようした画像処理手段を備えることを特徴とする。

【００２９】

さらに本発明は、この種のゲーム用画像処理装置において、前記モデルの予定された移動軌跡の終了点に対して、この終了点とは異なる目標点が設定され、前記終了点をこの目標点に一致させるように移動軌跡が補間される画像処理手段を備えることを特徴とする。

【００３０】

さらに本発明は、この種のゲーム用画像処理装置において、前記モデルが前記仮想空間内に置かれている高低差を求め、この高低差に応じてこのモデルに対して与えられている動作を補正する画像処理手段を備えることを特徴とする。

【００３１】

このように構成することにより、キャラクタ等仮想空間に設定されたモデルの動きをより現実感豊かに表示することができる。具体的には、一つのモデルが他のモデルの周りを周回する画像が容易に生成される。さらに、モデルが置かれているステージの状態をモデルの動きに反映させることができる。さらに、ステージの凹凸をモデルの投影像に正確かつ簡単に反映させることができる。さらに、モデルを遮るような位置に、壁等の構造物が仮想空間内に設定されていた場合でもこのモデルが遮られることなく表示される。また、

モデルのモーションを相手側のモデルの動き等に合わせて補正することができるので、モデルの多彩な動きを確実に表現することができる。さらに、モデルの高低差を考慮しているために、より現実感に優れた映像を生成することができる。

【００３２】

また、本発明の別の側面を成す一連の構成は以下ように開示される。その１つの具体的構成として、本発明は、仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置であって、モデルの移動軌跡を残像として表現するための残像表現処理手段を備え、この処理手段は前記モデルの現在モーション以前のモーションデータをそのまま記憶する記憶手段と、この記憶データを前記モデルの現在のモーションデータとともに表示する表示制御手段とを備えたことを特徴とする。この表示制御手段は、前記記憶手段のデータに演出を加えて表示するための演出手段を備えることができる。この演出手段は、半透明の画像処理を行うことができる。

【００３３】

別の具体的構成として、本発明は、仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置であって、前記モデルの移動特性を算出する算出手段と、前記仮想空間内に飛翔体を存在させるとともに、この飛翔体の一連運動を前記算出手段による算出結果に応じて制御する飛翔体制御手段とを備えることを特徴とする。 さらに別の具体的構成として、本発明は、仮想空間内に設定されたモデルの自由落下運動をシミュレートするゲーム用画像処理装置において、前記モデルの全体運動をシミュレートする第１の手段と、前記モデルに周回運動を付与する第２の手段とを備えることを特徴とする。

【００３４】

さらに別の具体的構成として、本発明は、仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置において、不定形の区画を前記仮想空間内に設定する区画設定手段と、この区画とこのモデルとの間に所定のベクトルを設定する手段であり、この区画を越えてこのモデルが移動しようとする場合には、このベクトルをモデルが区画を越えない方向に設定するベクトル設定手段と、このベクトルに応じて前記モデルの移動を制御するモデル移動制御手段と、を備えることを特徴とする。

【００３５】

さらに別の具体的構成として、本発明は、所定のパラメータを与えて周期的に変化するテクスチャーを形成する装置において、第１のテクスチャー列を作る第１の手段と、第２のテクスチャー列を作る手段とを備えるとともに、第１のテクスチャー列の最後或いは最初のテクスチャーと、第２のテクスチャー列の最初或いは最後のテクスチャーとが連続する形態になるようにそれぞれの手段にパラメータを与えるとともに、第１のテクスチャー列と第２のテクスチャー列の少なくとも一方の透明度を漸増或いは漸減して両方のテクスチャー列の最初の手テクスチャー同士から順に重ねて目的とするテクスチャー列を形成することを特徴とする。

【００３６】

また本発明は、周期的に変化するテクスチャーを形成する方法において、所定の方向に変化する第１のテクスチャー列を作る第１の工程と、同様に所定方向に変化する第２のテクスチャー列を作る工程とを備えるとともに、第１のテクスチャー列の最後或いは最初のテクスチャーと、第２のテクスチャー列の最初或いは最後のテクスチャーとが連続する形態になるようにするとともに、第１のテクスチャー列と第２のテクスチャー列の少なくとも一方の透明度を漸増或いは漸減して両方のテクスチャー列の最初のテクスチャー同士から順に重ねて目的とするテクスチャー列を形成することを特徴とする。

【００３７】
さらに別の具体的構成として、本発明は、仮想空間内に所定数のモデルを設定し、このモデルが前記仮想空間内で所定方向に移動するように制御されるとともに、所定位置の視点に基づくこの仮想空間の映像を表示手段に表示するようにしたゲーム用画像処理装置において、二つのモデル間に所定の条件が成立したか否かを判定する第１の手段と、前記条件が成立したことをそれぞれのモデルの運動に反映させる第２の手段と、この条件が成立したときに一方のモデルの運動の再生速度を他方のモデルに対して変化させる第３の手段とを備えることを特徴とする。
【００３８】
このように構成することにより、仮想空間でのモデルの動きをよりリアルに表現できるとともに、演出感豊かなゲーム画像を生成することができる。より詳細には、その１つの具体的構成によれば、ゲーム画像を形成する上で、演出効果を高めることの配慮があっても大きな計算負荷を与えることが無いゲーム用画像処理装置を提供することができる。また別の具体的構成によれば、このような負荷を少なくして、残像映像を実像画面と同時に表示することができるゲーム用画像処理装置を提供することができる。さらに別の具体的構成によれば、モデルの動きを正確に飛翔体に反映させることができるゲーム用画像処理装置を提供することができる。さらに別の具体的構成によれば、仮想空間内を自由落下する運動のシミュレートをより高品質なものにできるゲーム用画像処理装置を提供することができる。さらに別の具体的構成によれば、不定形の区画であっても、区画とモデルとの衝突判定肯定後のモデルの動き（モーション）の計算処理が的確且つ容易に実行できるゲーム用画像処理装置を提供することができる。さらに別の具体的構成によれば、画像処理を利用しても、繰り返して表示しても違和感無く連続映像が再現できる一連のテクスチャー列を作成できる。さらにまた、別の具体的構成によれば、より効果的な演出感を醸し出す上での再生速度の変化を可能にしたゲーム用画像処理装置を提供することができる。
【発明を実施するための最良の形態】
【００３９】
以下に、本発明の好適な実施例を、画像処理装置としてのビデオゲーム機について図面を参照して説明する。
【００４０】
第１の実施例
ビデオゲーム機の第１の実施例を図１～図３３に基づき説明する。
ハードウエアの説明
図１に、このビデオゲーム機のブロック図を示す。このビデオゲーム機によって、後述の各画像生成・処理１乃至６がそれぞれ実行される。
【００４１】
このビデオゲーム機は、装置全体の制御を行うＣＰＵブロック１０、ゲーム画面の表示制御を行うビデオブロック１１、効果音などを生成するサウンドブロック１２、ＣＤ－ＲＯＭの読み出しを行うサブシステム１３などによって構成される。
【００４２】
ＣＰＵブロック１０は、ＳＣＵ(System Control Unit)１００、メインＣＰＵ１０１、ＲＡＭ１０２、ＲＯＭ１０３、サブＣＰＵ１０４、ＣＰＵバス１０５等により構成されている。このブロックが本発明の画像生成装置の主体を担う。
【００４３】
メインＣＰＵ１０１は、内部にＤＳＰ(Digital Signal Processor)を備え、コンピュータプログラムを高速に実行できる。ＲＡＭ１０２は、ＣＤ－ＲＯＭを読み出すサブシステム１３から転送された各種ポリゴンデータを記憶する他に、メインＣＵ１０１のワークエリアとしても使用される。
【００４４】
ＲＯＭ１０３には、装置の初期状態において行う初期処理のためのイニシャルプログラムが格納される。ＳＣＵ１００は、バス１０５、１０６、１０７を介して行われるデータ

の転送を統括する。また、ＳＣＵ１００は、内部にＤＭＡコントローラを備え、ゲームの実行中に必要なる画像データをビデオブロック１１内のＶＲＡＭへ転送する。

【００４５】

コントロールパッド２ｂは、ユーザの情報入力手段として機能し、操作に必要な各種ボタンが備えられている。サブＣＰＵ１０４は、ＳＭＰＣ(System Manage r & Peripheral Cntroller)と呼ばれ、メインＣＰＵ１０１からの要求に応じ、コントローラ２ｂからペリフェラルデータを収集する機能を備えている。

【００４６】

メインＣＰＵ１０１は、サブＣＰＵ１０４から転送された周辺データに基づいて、ディスプレイに表示する画像の移動等の処理を行う。サブＣＰＵ１０４は、コネクタ２ａ（本体側端子）に接続した周辺機器の種類を判定し、判定した周辺機器の種類に応じた通信方式に従って、周辺データを収集する。

【００４７】

ビデオブロック１１は、図形生成手段として動作し、ポリゴンにより表示する画像を生成するためのＶＤＰ(Video Display Processor)１２０と、背景画のための画像合成、陰面処理、クリッピングを行うＶＤＰ１３０を備える。ＶＤＰ１２１及びフレームバッファ１２２、１２３へ接続される。

【００４８】

ディスプレイへ表示する仮想空間の画像を生成する際、表示に必要なポリゴンデータがメインＣＰＵ１０１からＳＣＵ１００を介してＶＤＰ１２０へ転送され、ＶＲＡＭ１２１に書き込まれる。ＶＲＡＭ１２１に書き込まれたポリゴンデータは、１画素あたり１６ビット又は８ビットの色情報を含む描画用データとして、描画用のフレームバッファ１２２又は１２３へ格納される。格納された描画用データは、ＶＤＰ１３０へ転送される。メインＣＰＵ１０１は、ＳＣＵ１００を介して、描画を制御する制御情報をＶＤＰ１３０へ供給する。ＶＤＰ１３０は、この制御情報に従って描画用データを処理する。

【００４９】

ＶＤＰ１３０はＶＲＡＭ１３１に接続されており、表示画面全体を上下左右に移動させたり、回転させたりするスクロール機能と、ポリゴンの表示順序を決定するプライオリティ機能（Ｚソート或いはＺバッファ）を備える。ＶＤＰ１３０は、描画用データをメモリ１３２を介してエンコーダ１６０へ出力する。エンコーダ１６０に出力された描画用データは、ビデオ信号のフォーマットに変換された後、Ｄ／Ａ変換されてモニタ装置５へ表示される。モニタ装置５には、このビデオ信号に基づいて画像が表示される。

【００５０】

サウンドブロック１２は、ＰＣＭ方式あるいはＦＭ方式による音声合成を行うＤＳＰ１４０と、ＤＳＰ１４０を制御等するＣＰＵ１４１により構成されている。ＤＳＰ１４０によって生成された音声データは、Ｄ／Ａコンバータ１７０により２チャンネルの信号に変換された後に２つのスピーカ５ａ又は５ｂに出力される。

【００５１】

サブシステム１３は、ＣＤ－ＲＯＭドライブ等を搭載し、ＣＤ－ＲＯＭ等の記録媒体によって供給されるアプリケーションソフトの読み込み、動画の再生等を行う機能を備える。

【００５２】

以上説明した画像生成装置によって行われる処理について説明する。図２は、ＣＰＵブロック１０によって実行されるメイン処理を示すフローチャートであり、キャラクタを円運動させるための円運動処理Ｓ２００、仮想的摩擦処理Ｓ２０２高低差がある地面への影の表示（Ｓ２０４）：すわなちキャラクタの投影表示処理、交差判定処理（Ｓ２０６）、キャラクタのモーション処理制御（Ｓ２０８）、キャラクタの高低差処理（Ｓ２１０）が順次行われる。必要が無い処理は、行われること無く次の処理に移行する。各処理の内容を以下に詳説する。

【００５３】

円運動処理

この処理は、概ね次のような内容である。既述のコントロールパッド２ｂの十字状に形成された方向キー２６ｂ（図２６をも参照）を図示左又は右に操作すると、この方向キーによって操作されるキャラクタ（闘士）３０（図３参照）が仮想空間内を上下左右方向に移動する。円運動処理モードでは、遊戯者が操作する闘士３０と相手側の闘士３２との間に仮想的に向心力（相手側のキャラクタに向かう力）が与えられ、遊戯者側の闘士３０が相手側の闘士の回りを自動的に周回（円運動）する処理３０が実現される。

【００５４】

この処理内容を、図４のフローチャートに基づいて説明する。ステップ４００において、円運動処理モードに入った否かが判定される（判定１）。遊戯者（プレーヤ）が所定の操作を成した場合に円運動要求があったとされる。例えば、コントローラ（パッド２ｂ）の所定の釦を押下した場合である。次に、図２６の２６ｂの方向キーの図示左右方向操作量が読み込まれる等して、遊戯者の闘士と相手側の闘士との位置とが検出され、相手側の闘士が遊戯者側の闘士に対して成す方向が計算される（ステップ４０２）。

【００５５】

次いで、遊戯者側の闘士３０にその方向の角度が与えられて、かつ、この闘士を相手側の闘士に向かせる仮想的な向心力（求心力）が与えられる（ステップ４０４）。この求心力を、物理的に考えれば、求心力は２つの物体間に働く引力に等しい。

【００５６】

相手側の闘士のｘ座標とｚ座標とをそれぞれ（ｅｘｐ，ｅｚｐ）とし、相手側の闘士のｘ座標とｚ座標とをそれぞれ（ｍｘｐ，ｍｚｐ）とすると、既述のような相手側闘士３２の方向は、ａｒｃｔａｎ（ｅｘｐ－ｍｘｐ，ｅｚｐ－ｍｚｐ）によって計算される。なお、この計算に際して、両方の闘士の高さ方向のｙ方向の座標は、便宜上考慮していない。

【００５７】

この結果、求心力と横方向キーによる左右方向の速度によって、仮想空間内の円運動が作られる（ステップ４０６）。この円運動によって遊戯者の闘士３０が相手側の闘士３２を向きながらその回りを回るように動く。図５は、この円運動を仮想空間の上方より見た概念図であり、闘士３０が状態１から状態２、状態３を経て状態４までの円運動を行う。既述の図３は、状態１に係わる仮想空間を所定の視点から表した映像であり、既述のモニタ５（図１参照）に表示される映像である。図６は、状態４に係わる、図３と同様な映像である。

【００５８】

したがって、説明された円運動処理によれば、一つのキャラクタに仮想的な向心力を与えているために、遊戯者にとっては、図２６の方向キー２６ｂを図示左右方向に操作するだけで、容易にキャラクタ３０に円運動を与えることができる。例えば、既述のように相手側闘士３２の回りを自己が操る闘士３０が周回する動きを簡単に実現できる。

【００５９】

仮想的摩擦処理

この処理は、キャラクタの動きに、地面の傾きによる滑りと、地面の性質によってキャラクタに及ぼす摩擦力を変え、かつ、キャラクタが移動しているときには動摩擦を与え、キャラクタが移動していないときには静摩擦を与えることにより、キャラクタの動きをより多彩にすることが内容である。

【００６０】

先ず、図７は、仮想空間内の地面の断面を示したものである。図８に示すように、ステップ８００において、地面の傾きから滑り量を次のようにして計算する。

滑り量（＝ｄｖ／ｄｔ）＝ｉｖｇ

ｉ：所定の定数

ｖ：地面の傾きを表す法線単位ベクトル

ｇ：重力加速度

ここで、法線単位ベクトルとは、図７（１）に示すような法線ベクトルの地面の接線方向

へのベクトルである。
【００６１】
この法線単位ベクトルが仮想空間の座標軸（ｘ軸）に対して成す角（θ）が大きくなると、この地面上に在るキャラクタに対して、傾いた地面に沿って下がろうとする滑り量（加速度）がより大きく与えられる。
【００６２】
次のステップ８０２においては、キャラクタ（闘士）が移動しているか否かが判定される。例えば、遊戯者側から云うと、闘士の動きを制御するための方向キーが操作されているか否かが判定される（判定１）。
【００６３】
キャラクタが動いている場合は、動摩擦が与えられ、キャラクタが動いていないときは、静止摩擦が与えられる（ステップ８０４，８０６）。摩擦（＝ｄｖ／ｄｔ）自体は、次のようにして計算される。
Ｍ・（ｄｖ／ｄｔ）＝μ・Ｓ・Ｍ（ｇ・ｃｏｓθ＋（ｄｖ'／ｄｔ）・ｓｉｎθ）
ｄｖ／ｄｔ＝Ｔ・ｇ・ｃｏｓθ＋Ｔ・（ｄｖ'／ｄｔ）・ｓｉｎθ）
Ｍ：重さ
ｖ：摩擦速度（計算後の物体の速度）
ｖ'：物体の速度
μ：摩擦係数
Ｓ：キャラクタの接地面積（定数）
ｇ：重力加速度（定数）
θ：地面の角度
Ｔ＝μ・Ｓ
【００６４】
動摩擦及び静止摩擦は、この式によって得られた摩擦（或いは摩擦係数）に所定の処理を与えることにより決定される。但し、静止摩擦は動摩擦より大きな値となる。この摩擦は地面の角度によって変化するものであり、地面の傾きが大きいほど摩擦は小さい値となる。
【００６５】
次に、地面の属性が決定される。すなわち、キャラクタが接地している地面が、例えば、水場或いは砂場であるかが判定される。これらの属性は、予め地面毎に仮想的に与えれている。例えば、砂場では計算された摩擦量を２倍とし、水場では摩擦量を３倍する。次いで、先に計算された滑り量に、この摩擦力を減じるように反映させる（ステップ８１０）。
【００６６】
この処理によって、静止摩擦あるいは動摩擦がキャラクタに常に与えられるために、止まっているキャラクタはなかなか動き出せず、一度動き始めるとより移動し易くなる。また、地面の属性によってキャラクタの移動特性を変えることができる。これらのことにより、キャラクタの移動をより現実感豊かに表現することが可能となる。
【００６７】
高低差がある地面への影の表示：
図９にあるように、この処理はキャラクタ９１が立っている地面９０に凹凸９２がある場合、キャラクタの影（投影像）をこの地面に凹凸をつけながらより簡単な計算によって表示することを内容とする。図９はこの処理の概要を示す。図中の９４は高さ０の仮想的な基準線であり、そして、９６は無限遠にある平行光源である。
【００６８】
この人物は、頭、胸、腹、腰、足等のオブジェクトの集合構成されている。各オブジェクトは、複数のポリゴンによって作られている。図面のｙＧは基準線からキャラクタが立っている地面迄の距離であり、ｙ１・・・ｙｎは、各オブジェクト毎の地面から基準線までの距離である。 この処理は、次の計算式によって演算される。

Ｅ'（ｘ', ｙ', ｚ'）＝Ｍｓ・Ｅ（ｘ，ｙ，ｚ）
Ｅ'（ｘ', ｙ', ｚ'）：影の座標点
Ｅ（ｘ，ｙ，ｚ）：人物（キャラクタ）の座標点
Ｍｓ：影の座標をワールド座標系に変換するためのマトリックス（行列式）である。
【００６９】
このＭｓは次のようにして与えられる。
Ｍｓ＝Ｍｕ・（－Ｔｙｇ）・Ｐｎ・Ｔｙｎ・（－Ｔｓｎ）・Ｒｎ・Ｔｓｎ
Ｍｕ：人物の座標を図９のワールド座標系に変換するための変換マトリックス
Ｔｙｇ：人物が立っている地面から基準線まで平行移動させるための平行移動マトリックス
Ｐｎ：人物を基準線に斜投影するための変換マトリックス
Ｔｙｎ：斜め投影された影を各オブジェクトの接地点（即ち、各オブジェクトが置かれる地面）迄平行移動させるためのマトリックス
Ｔｓｎ：各オブジェクトをワールド座標系の原点に対して平行移動させるための平行移動マトリクス
Ｒｎ：地面の傾きに合わせて影のポリゴンを回転させるためのマトリクス
【００７０】
この処理は、次の過程によって行われる。処理過程１：影を作るための人物類似のキャラクタ９１が地面から高さ０の基準線まで平行移動され、次いで、処理過程２：この人物類似のポリゴン集合体９１を、前記平行光源９６に基づいて基準線９４に斜投影し、次いで、処理過程３：斜め投影されたオブジェクトを各オブジェクト毎にｙ１・・・ｙｎの幅で地面までそれぞれ平行移動させ、次いで、処理過程４：各オブジェクトをワールド座標系の位置からその系の原点まで平行移動させ、次いで、処理過程５：それぞれの地面の角度に応じて、各オブジェクトを回転させ、最後に、処理過程６：処理過程４によって原点まで移動したオブジェクトを元の座標系に戻すための平行移動が行われる。
【００７１】
したがって、この処理によれば、地面９０に凹凸がある場合、この凹凸に合わせてキャラクタの投影像を凹凸を付けて簡単に地面９０上に表示することができる。
【００７２】
交差判定処理
図１０は、二つのキャラクタ、例えば、互いに向き合う闘士（人物１，人物２）を把える仮想カメラの視野領域を示すｘｚ平面図である。この仮想カメラ（視点）によって既述の図３に示すような映像が表示される。この処理においては、この視野領域に闘士以外の壁、建屋等視野領域を遮るような仮想の構造物が重なった場合、この構造物の一部或いは全体を表示しないようにされる。図１０において、構造物１００，１０２は、視点から人物に至る視野領域を遮るのに対して、構造物１０４は視点によって表示される領域にあるものの前者の領域を妨げない。ここでは、構造物１０２と１０４とを消去する画像生成或いは画像処理が行われる。
【００７３】
図１１は、この処理を示すフローチャートである。この処理に際して、これらの構造物が視野領域に重なっているか否かの判定をより迅速かつ容易に行うために、ｘｚ座標面において判定が行われ、かつ厚さを持ったものは、円によって近似される。図１２は、この構造物を内接円で近似した状態を示したｘｚ平面図である。
【００７４】
図１１において、ステップ１１０は、この近似円が前記視野の領域に重なっているか否かを判定する処理である。図１３はこのステップにおける判定の概念図である。視点から人物１，２に至るまでの視野領域に相当する三角形の領域を構成する全ての辺について、中心点Ｔの半径ｄを持つある近似円が重なるか否かの判定を以下の処理に基づいて行なう。なお、以下の説明において、ベクトルＬ、ベクトルＲ、ベクトルＴは、所定の原点からＬ、Ｒ、Ｔに対して成すベクトルである。なお、三角形の各辺のそれぞれの頂点について

時計方向にＬ点、Ｒ点が順に設定される。

【００７５】

【数１】

$$\vec{r} = \vec{R} - \vec{L}$$

$$\vec{t} = \vec{T} - \vec{L}$$

$$\vec{p} = \vec{c} + \vec{L} - \vec{T}$$

$$cx = rx \cdot \frac{|\vec{c}|}{|\vec{r}|} \quad , \qquad cy = ry \cdot \frac{|\vec{c}|}{|\vec{c}|}$$

【００７６】

ここで、

【数２】

$$|\vec{c}| = |\vec{t}| \cos\theta$$

ｃｘ：ベクトルｃのｘ成分
ｃｙ：ベクトルｃのｙ成分
ｒｘ：ベクトルｒのｘ成分
ｒｙ：ベクトルｒのｙ成分

【００７７】

内積の定理により、

【数３】

$$|\vec{r}| = |\vec{t}| \cos\theta = rx\ tx + rz\ tz$$

$$\frac{|\vec{c}|}{|\vec{r}|} = \frac{rx\ tx + rz\ tz}{|\vec{r}|^2} = \frac{rx\ tx + rz\ tz}{rx^2 + rz^2}$$

【００７８】

但し、

【数４】

$$0 \leqq \frac{|\vec{c}|}{|\vec{r}|} \leqq 1$$

【００７９】

近似円の中心Ｔが直線ＴＲの正領域にある場合（三角形の内側より外にＴがある場合）、

【数５】

$$|\vec{p}| - d > 0$$

【００８０】

ならば、円は三角形の内側の領域に対して独立、すなわちこの領域に重なることはなく、

【数６】

$$\frac{|\vec{c}|}{|\vec{r}|} < 0 \quad \text{または} \quad 1 < \frac{|\vec{c}|}{|\vec{r}|}$$

の時は、点Ｔが直線ＬＲの正領域にあれば、円は領域に対して独立となる。

【００８１】

ステップ１１０において、近似円が三角形の領域内に重なっている場合は、この構造物が表示されないように処理され（ステップ１１１）、重なっていない場合は、次の処理（ステップ１１２）に進む。次の処理は、この構造物が一定方向に長さを持った壁の如くのものについての重なり判定を行うことを内容とする。この処理は、図１４に示すように次の如く行われる。

【００８２】

直線ＴＳとして近似された構造物について、この直線と三角形の領域との交差（重なりの一種）を判定する。三角形の領域を構成する全ての辺について、以下の処理を行う。直線ＬＲと任意の点Ｐとの位置関係を判定する関数をＦ１（Ｐ）とすると、

【００８３】

【数７】

$$F1(P) = a1Px + b1Pz + c1$$
$$a1 = Lz - Rz$$
$$b1 = Rx - Lx$$
$$c1 = LxRz - RxLz$$

【００８４】

直線ＳＴと任意の点Ｐとの位置関係を判定する関数をＦ２（Ｐ）とすると、

【数８】

$$F2(P) = a2Px + b2Pz + c2$$
$$a2 = Sz - Tz$$
$$b2 = Tx - Sx$$
$$c2 = SxTz - TxSz$$

Ｆ１（Ｓ）・Ｆ１（Ｓ）＜０かつ、Ｆ２（Ｓ）・Ｆ２（Ｓ）＜０ならば、直線ＴＳは直線ＬＲに交差していると判定される。

【００８５】

この交差が否定された時は、ステップ１１４に移行し近似円の重なりの有無が判定される。この理由は次のとおりである。三角形の領域内に既述の直線ＴＳが完全に入ってしまった場合、重なり判定が否定されて壁状の構造物が表示されてしまう。そこで、この直線ＴＳに図１５のような壁の厚さに相当する小円１５０を与え、この小円についてステップ１１０と同じ様な重なり判定を行う。この結果、直線ＴＳが完全に二角形の領域内に入ってステップ１１２の判定において重なり無しと判定された場合でも、この近似円が三角形の領域内に入っている場合、重なりあり、と判定されてステップ１１１に移行する。

【００８６】

これらの処理により、仮想カメラから表示対象となっているキャラクタに向けた視野領域に壁、柵、塀等の構造物が表示されないようできる。そして、図１０に示すように、この壁等の構造物がこの視野領域から出ている１０４のような場合は、そのままこの構造物が表示される。以上の処理によれば、余計な構造物によってキャラクタを遮ることなく好適にキャラクタを表現することができる。もっとも、表示対象としてのキャラクタが設定されていない場合は、これらの構造物はそのまま表示される。なお、近似円としては、図１２の場合の内接円に代えて外接円でも良い。

【００８７】
キャラクタのモーション処理制御
ここでの処理は、一つのモーションから他のモーションをつないで一連に表示させようとする場合で一つのモーションの終了位置を目標点に合わせるように補間することを内容とする。
【００８８】
図１６は一つのキャラクタのモーションの一連の過程を示すものであり、人物類似のキャラクタ（闘士）１６０がプレイヤによるパッド２ｂの操作によって相手側の闘士に向けて、腕部１６０Ａによる殴打を繰り出す過程が示されている。
【００８９】
この腕部の挙動は、公知のスプライン関数（特に、３次スプライン関数）によって計算されモニタに表示される。今ここで、この腕部１６０Ａが相手側闘士の胸部に向けて移動しているとして、ある時点で相手側闘士がｙ方向に移動した場合（例えば急に屈んだ場合）、遊戯者にとってみれば、移動している腕の軌跡を急激に図１７の如く下方に変えることは非常な困難を伴うし、また、このことをＣＰＵブロック１０の側で単に行おうとすると腕の動きが不自然なものになり易い。
【００９０】
そこで、現在実行中のモーションのスプライン軌跡変化過程の終了付近につなぎ領域を設け、このつなぎ領域に、当初のスプライン軌跡の終了点が別な位置の目標点になるように、新たなスプライン関数を与えるなどの補間を行うようにした。
【００９１】
このことを図面を用いて説明する。図１８の（１）に示すように、図１６の闘士の腕部の先端にある手の甲１８０Ａは、スプライン関数によって定められた軌跡１８０に沿って移動する。図１８の（２）は、このスプライン関数によって得られた軌跡１８０を示したものであり、手の甲１８０Ａはこの軌跡の開始点から移動して終了点に至り、その後開始点まで復帰する。
【００９２】
今、手の甲１８０Ａがスプライン関数軌跡１８０の開始点からその終了点に向けて移動しているとする。この過程で、例えば相手側闘士の挙動変化等によって当初の終了点を目標点に変更する必要が生じたとする。
【００９３】
そこで、ＣＰＵブロック１０は、手の甲の位置４８０Ａがつなぎ領域にある時点で目標点において終了するように、この領域内の関数を補正する。この補間は、例えば、後述のように行われる。勿論、つなぎ領域のスプライン関数の次数や係数等を適宜変えることによっても可能である。その結果、図１６に示すモーションが図１７に示すモーションに自動的に補間されるし、つなぎ領域におけるモーションもより円滑に表現される。なお、図示する上での便宜上、図１８の（２）において縦軸はｙ軸方向の移動量のみを示している。３次元ビデオゲームの一般論のとおり、ｘ軸とはモニタの画面に臨んで横方向に与えられ、ｙ軸はその高さ方向に与えられ、ｚ軸はその奥行き方向、すなわち画面と直交する方向に与えられる。
【００９４】
つなぎ領域（つなぎ時間）はエンドフレームを備えた複数のフレーム群によって適宜設定される。また、手の甲以外の上腕部及び下腕部の各オブジェクトについても、同様な処理が適用されて腕全体の補間されたモーションが完成される。つなぎ領域における補間された座標は、次の特性式によって設定される。
現在座標＋（現在座標－目標座標）・（現在時間／つなぎ時間）
【００９５】
図１９は以上の処理を説明するためのフローチャートであり、ステップ１９０において、遊戯者がパッド２６ｂを操作してモーション移動のコマンドが発生したか否かが判定される。ステップ１９２では、終了点と異なる座標に目標点が設定されたか否かが判定され

る。ステップ１９４では、選択されたモーションが再現される。ステップ１９６では、この判定が肯定された場合、つなぎ時間が計算されるとともに、ステップ１９４に移行しこのつなぎ時間の間補間されたモーションが再現される。ステップ１９８では、現在のフレームが一連のモーションのエンドフレームか否か判定され、エンドフレームである場合には、メインルーチンにリターンし、エンドフレームに未だ至っていない場合には、ステップ１９２に戻る。

【００９６】

ここで、つなぎ時間を成すつなぎフレームはモーションの最大フレーム数（図１８（２）のスタートフレームからエンドフレームまでのフレーム数）の０．１であり、つなぎフレームが１０を越える場合は１０に固定される。つなぎフレームは、１以上、最大フレーム数の１／２の範囲であれば良い。つなぎフレームが余り長いとモーションが持っている元々の動きが失われ、一方、余り短いと補間が急峻となり動きに円滑さが失われる。

【００９７】

この処理によれば、次のような効果が達成される。ビデオゲーム装置のように、ゲーム画面に登場する闘士等のキャラクタを素早く操作するためにコントローラが操作されると、キャラクタの多彩な動きが連続的に再現される。このとき、一連のモーションが再現されている時点で、相手のキャラクタの急な動作に合わせてそのモーションを変化させようとしても、モーション自体が固定されたものである限りそれは困難である。しかしながら、既述のようにモーションが補間されると、例えば相手側キャラクタの動きに合わせたモーションを多彩に再現できることになり、モーションがより現実感豊かなものになる。通常、このような補間をユーザが行うことは極めて困難であり、なによりも、従来のビデオゲーム装置ではこのような補間を行うようには構成されていない。

【００９８】

なお、この処理は、スプライン関数によってモーションが計算される場合を例にとって説明したが、予め決められたパターンが順次再現されるパターンチェンジの方式に、この処理を適用することも可能である。この場合は、つなぎ領域において補正されたパターンを再現するようにすれば良い。

【００９９】

キャラクタの高低差処理

この処理はキャラクタ（闘士）に互いに地面からの高低差があるときに、この高低差を補正して所望の画像を表示することを内容とする。例えば、図２０に示すように、互いに向き合う二人の闘士２００，２０２との間に高低差が存在する場合、闘士２００からパンチを高い位置にある闘士に向けて水平に繰り出すことは現実的ではない。そこで、この処理は、この高低差を補正して、闘士２００からの殴打を高い方向に向けて繰り出すようにさせた（図２３参照）。

【０１００】

この処理は、図２１に示すように、ステップ２１００において、この高低差を補正する必要があるか否かが判定される。この判定において「必要あり」とされるのは、例えば、図のように互いに向き合う闘士について、一方の闘士から他方の闘士に対して、腕や足による殴打の攻撃要求（コマンド）が発生した場合である。

【０１０１】

次に、この判定が否定された場合は、メインルーチンに復帰する。一方、この判定が肯定された場合は、さらに、ステップ２１０２において、相手側闘士が地面上に立っているか、すなわち、キャラクタと地面との衝突判定が行われ、これが肯定された場合には、キャラクタの高低差処理を必要として次にステップに移行する。この判定が否定された場合は、図のルーチンを終了する。

【０１０２】

次のステップ２１０４では、遊戯者自身が操るキャラクタが立っている地面からのこのキャラクタまでの距離を演算する。相手側のキャラクタについても同じ演算を行う。次に、ステップ２１０６において、この結果の値について所定値と比較し、所定値の範囲を越

える場合は、ステップ２１０８でこの範囲内に前記距離に関する差分を収める処理を行う。すなわち、一方の遊戯者が操る側の闘士の高さ（ｍｙ）と他の遊戯者が操る側或いは画像処理装置自体が所定のプログラムによって自動的に操る側の闘士の高さ（ｅｙ）とすると、両者の差分（ｄｉｆｆ１）は、（ｍｙ－ｅｙ）となり、例えば、ｄｉｆｆ１が、－０．１５≦ｄｉｆｆ１≦０．１５の範囲内か否かが判定され、ｄｉｆｆ１がこの範囲を越える場合は、この範囲より小さいｄｉｆｆ１は、これを－０．１５とし、この範囲より大きいものは、これを０．１５とし、この範囲内である場合は、ｄｉｆｆ１をそのまま差分結果（ｄｉｆｆ２）として決定する。このようにｄｉｆｆ１に対して補正を行う理由については後述する。

【０１０３】

次のステップ２１１０では、ｄｉｆｆ２を攻撃力が発生する時間（フレーム数）で割り、この結果を「結果１」とし、ｄｉｆｆ２を攻撃力発生から硬化が解けるまでの時間で割ってこの結果を「結果２」とする。なお、このことを図２２を用いて説明する。図２２は、攻撃技（闘士が繰り出す蹴り、パンチ、ジャンプ等）の流れを示したものであり、１，２・・・５の各時間毎のフレームの流れ（時間の流れ）に対して次のようである。

１：技の開始（０フレーム目）
２：攻撃力の発生（１から３フレーム目迄）
３：攻撃力の消滅（３から４フレーム目迄）
４：硬化時間（これは他の攻撃技のコマンドを受け入れない時間、０から４フレーム迄）
５：技の終わり（０から５フレーム目迄）

【０１０４】

したがって、既述の「結果１」は、（ｄｉｆｆ２）／２となり、「結果２」は、（ｄｉｆｆ２）／（４－２）となる。これらの値は、当然のことながら、所定のワークＲＡＭ領域に設定記憶される。図２２に示す特性は各技毎に予め決められている。結果１は攻撃力発生迄のフレーム毎の変化率に相当する値であり、結果２は、硬化が解ける迄のフレーム毎の変化率に相当する値である。

【０１０５】

次いで、ステップ２１１２では、要求された技毎に技を繰り出す際の手や脚等が移動する座標を決定する。この座標値は、相手側のキャラクタに対して高さの差が無いことを前提に予め決められている。この座標（リーフ）を「結果３」とする。このリーフは、リーフを返すための関数ｆａｔによって技を繰り出す際の手や脚の空間座標位置に応じて決定される。次のステップ以降は、攻撃技を実行する際の処理を説明するものである。

【０１０６】

ステップ２１１４において、技が発生してから現在表示されるフレームが何フレーム目に相当するかが検出される。次のステップ２１１６では、このフレームが攻撃力が発生する以前のフレームであるか否かが判定される。図において、フレーム１迄は攻撃力が発生する以前のフレームであると判定される。

【０１０７】

次にステップ２１１６における判定が肯定された場合、ステップ２１１８において、前記「結果１」とフレーム数を乗じ、この演算結果を差分として前記ステップ２１１２のリーフに加算する。そして、この結果を用いて公知のインバースキネマテックの技術（例えば、「ロボットの力学と制御」、システム制御情報学会編、有本卓著、朝倉書店発行、５．２ 逆運動学（１３２頁～））を用いて、リースにこの差分が加えられた値等に基づいて闘士の体の形を再計算する。

【０１０８】

既述のステップ２１１６において、攻撃力が発生した以降であると判定された場合は、ステップ２２００において、硬化が解けるフレーム数（図の「４」）から現在のフレーム数（図の「２又は３」）を減じ、結果２にこのフレーム数を乗じてステップに移行する。

【０１０９】

この結果、ステップ２１１８によって図２３に示すように自己が操る闘士２００から相

手の闘士２０２に向けて、闘士の高低差を補償した高さにパンチを繰り出し、パンチの命中後或いはパンチが命中しないとしても、その後パンチとして繰り出された腕を高低差を補償しながら元の位置まで帰還させる。

【０１１０】

ここで説明した処理によれば、二つの闘士の高低差を反映させた画像処理が実行されるので、一つの闘士から繰りだした攻撃（パンチ、キック）が高い位置に在る闘士に向けて行われ、実際の攻撃の現状に即した映像を提供することが可能となる。したがって、高い位置にある闘士に対するモーションを事前にメモリに記憶していなくても、一つのキャラクタと他のキャラクタとの間の画像処理を、キャラクタ間の高低差を反映した状態で実現することができる。

【０１１１】

なお、ｄｉｆｆ１に対して補正を行ったのは、次の理由による。このような袖正を行わないと、両方の闘士の間に大きな高低差があると、一方の闘士から他方の闘士に向けた殴打が、極端に云うと地面に対して直角方向に向けられてしまい、かえって不自然になるからである。

【０１１２】

なお、既述の各処理の説明において、ビデオゲームを例にとり説明したがこれに限定されるものではない。図１において、ＣＤ－ＲＯＭに代えてＲＯＭカセットを用いても良い。これらのものは本発明に係わる画像生成装置の動作プログラムが記憶された記憶媒体として機能するものである。また、交差判定処理において、既述の構造物を完全に表示しない場合の他、この構造物をメッシュポリゴン或いはポリライン等で表示することも、「表示しない」の態様に含まれることとする。なお、図２４及び図２５は、本発明のモデルとしての人物類似のキャラクタ（闘士）のそれぞれ一例である。

【０１１３】

ＣＰＵブロック１０によるグローシェーディング（その他のシェーディングについても同様）に当たっては、キャラクタの全体ポリゴンについて行う場合のほか、キャラクタの一部、特に色を線形補間して立体的に見せたい部分（例えば、露出している皮膚に相当するポリゴン）のみについて処理を行うこともできる。こうすることにより、ＣＰＵブロック１０に与える処理遂行上の負荷を低減することが可能となる。図２４のキャラクタでは衣服より露出している手の甲、顔、そして首下の胸部についてのみグローシェーディングが行われ、図２５のキャラクタでは、ほぼ上半身並びに脚部についてのみグローシェーディングを行うようにしても良い。

【０１１４】

また、各ポリゴンに対するＺソートをかける場合、４角形のポリゴンでは従来より４頂点を利用していたが、これを２頂点をそれぞれ結ぶ対角線の交点（中点）のみで判定するようにしても良い。この結果、メインＣＰＵがメモリをアクセスする頻度を減少させて処理速度を向上させることができる。なお、前記中点によってＺソートをかけてもこの中点は４頂点の平均値であるため、４頂点を用いたＺソートの精度をそのまま維持することができる。三角形のポリゴンの場合は、重心を利用すれば良い。

【０１１５】

次に、既述の画像生成装置の処理動作の他の実施例について説明する。図２６は、前記コントロールパッド２ｂの詳細を示す正面図であり、このものは、既述の方向キー２６ｂとＡ、Ｂ、Ｃ、Ｘ、Ｙ、Ｚ、Ｌ、Ｒの各ボタンを有している。これらの方向キーやボタンの押下は、図２４或いは２５のキャラクタ（闘士）の動きに対応しており、例えばＡボタンは「相手闘士からの攻撃に対する防御行動」、Ｂボタンは「相手側闘士に繰り出すパンチ」、Ｃボタンは「相手側闘士に向けて繰り出すキック」に相当する。

【０１１６】

ユーザである遊戯者はこれらのキーやボタンを多彩に操作して自己が望むようにキャラクターを操作しようとするが、極めて迅速かつ多彩に動く闘士を適切に操作するには非常な熟練を要する。

【０１１７】
そこで、ここで説明する処理の形態においては、図２７に示すように、複数のボタンを押下することの組み合わせであるコマンド技の集合を混合技とし、これをファイルとして予めワークＲＡＭに設定する。そして、これを既述のボタンのうちの一つを押下することによって読出し、闘士がコマンド技を順に繰り出すようにしている。図２７において、ＰＰＫのようにあるのは、Ｂボタン、Ｂボタン、Ｃボタンが順に押下された場合に等しく、Ｐ＋Ｋのようにあるのは、ＢボタンとＣボタンが同時に押下された場合に等しい。さらに、各コマンド技についてその技が占めるフレーム数が設定できるようになっている。したがって、遊戯者は混合技中の各コマンド技のフレーム数を自由に設定できるために、自己が望む混合技を多彩に作り出すことができる。この混合技の具体例を述べれば、「相手側の闘士を巴投げし、起き上がったところに蹴りを与える」等である。

【０１１８】
さらに詳細に説明すれば、「Ｐ、Ｋ、Ｐ」は、パンチボタン、キックボタン、パンチボタンが連続して押されることを示し、且つ、これらが、連続されたと判断されたときは巴投げをすると仮定すれば、Ｐが押されたときに、キャラクタはパンチのモーションに入る。このモーションの始まりから、例えば、モーションがパンチ戻りになるまでの間に、Ｋが押されたとすれば、繰り出されたパンチの姿勢から、パンチ戻りのモーションになるタイミングで、キックが始まり、キックのモーションが繰り出される間にさらに、Ｐが入力されたときは、連続技と判断して、巴投げのモーション（コマンド枝）に連続していく。この時のタイミングを、先のキックで、相手のキャラクタが倒れかかっているタイミングに合うようにすれば、同じ巴投げでも、技が掛かり易い等、非常に効果的な巴投げを作ることができる。

【０１１９】
遊戯者は、これらの混合技ファイルを複数作り、各ファイルに一つのボタンを割り付けることができる。もっとも、この混合技ファイルが割り付けられていないボタンを押下すれば元もとこのボタンに割り付けられている単独技が繰り出される。どの範囲のボタンに混合技を登録するかは、混合技の必要性或いは単独技の必要性に応じて適宜決定される。各ファイルに割り付けるボタンは、複数であっても良い。

【０１２０】
次のこの処理の動作を図２８に示すフローチャートに基づいて説明する。先ずステップＳ２８０において、既述のボタンが押下されたか否かが判定される。ポタンが押下された場合、押下されたボタンに対して混合技が登録されている場合（ステップＳ２８２肯定）、ステップ２８４において、ワークＲＡＭに登録されている混合技中の各コマンド（複数或いは単独のボタンの押下に等しい）が順に読み出される。次いで、ステップＳ２８６において、ファイル中のコマンドが最後まで実行されたか否かが判定され、これが肯定された場合はこのフローチャートが終了されるまで繰り返し実行される。

【０１２１】
一方、ステップＳ２８０においてボタンの押下が無い場合はリターンし、また、ステップＳ２８２において押下したボタンが混合技のファイルに登録されていない時には、そのボタンに割り付けられた単独技が実行される。

【０１２２】
ここで説明した形態によれば、キャラクタ（モデル）に対する動作コマンドの組み合わせ（例：動作スイッチを押下する組み合わせ：一つのスイッチの押下、複数のスイッチの押下、これらスイッチを順に押す、あるいてはこれらスイッチを同時に押す）を予めメモリに記憶させ、これを簡単なスイッチ操作、例えば、一つのボタンを押下すること、によって、画像処理手段が読み出し、読み出されたコマンド群に基づいてモデルの動き（モーション）を連続的に制御するようにしている。したがって、既述のように、ユーザが複雑なキー操作を行うことなく、かつ自己の好みに応じたより多彩なモデルの動き（モーション）を表現することができる。

【０１２３】

なお、図２９は、ここで説明した混合処理の際に、モニタに表示されるモードセレクト（モード選択）のための初期画面であり、混合技の選択のためには上段右端のアイコンが選択される。図３０は、ボタン（キー）を技に割り当てた(Assign)したことを示す画面、図３１は、前記混合技ファイルを作成するための画面、図３２は、互いの闘士が格闘状態にあることの１状態を示す画面である。図３３はこの１状態の画面にさらに後に展開される画面であり、闘士（「あきら」）が闘士（「ラウ」）に倒された状態のものである。なお、図３３から明らかなように、一方の闘士が他方の闘士を倒したとき（ＣＰＵブロック側でそう判定されたとき）、画面中のキャラクタ３３０がそれを表示するようになっている。図３３においては、「つぼみの植物類似のキャラクター」が「花がさいた植物類似のキャラクター」に変更されることによってこのことを表示するようになっている。

【０１２４】

第２の実施例

続いて、画像処理装置としてのビデオゲーム機の第２の実施例を図３４～図６８に基づき説明する。なお、この第２に実施例におけるビデオゲーム機のハードスエア構成は前述した第１の実施例のものと同一であるので、その説明は省略する。

【０１２５】

図３４は、ＣＰＵブロック１０によって実行されるメイン処理を示すフローチャートであり、後述の各処理が実行される。Ｓ２００は残像処理ルーチンであり、Ｓ２０２は舞上げ処理のルーチン（飛翔体の処理）であり、Ｓ２０４はモデルの自由落下運動処理、Ｓ２０６は不定形区画とモデルとの衝突判定処理、Ｓ２０８は再生速度の変更処理を行う。これらの各処理は、繰り返し実行される。以下、各処理について説明する。

【０１２６】

残像処理

図３５は、モデルとしての闘士に対して適用される残像処理の原理を示す。図３５において、３０は闘士であり、闘士の脚３０Ｂについて残像が表示されている。すなわち、現在のフレーム（コマ）の実像が表示されると同時に、以前のフレームの残像１乃至４が同時に表示されている。

【０１２７】

図３６はこの残像処理の原理を示すものである。ポリゴンデータファイル（或いはパラメータファイル）は、この闘士の脚の「廻し蹴り」モーションを規定するパラメータを記憶している。ポリゴンデータファイルの＃１乃至＃ｎは、残像開始時点からのコマ数を示している。

【０１２８】

ポリゴンデータファイル＃１には、残像４に相当するモデル＃１が記憶されている。従って、この時点では、画面上に闘士のモデル＃１が実像とし表示される。次の時点では、ポリゴンデータファイル＃２に、残像３に相当するモデル＃２が記憶される。したがって、この時点では、画面上に闘士のモデル＃２が実像として表示される。以後同じようにして、画面上に闘士の脚モデル＃３・・・・、＃ｎが表示される。図４の４コマ目の場合（＃４）は、脚の実像モデルとしては、＃１乃至＃４までが順次表示される。

【０１２９】

残像用バッファとして、Ｂ＃１、Ｂ＃２、Ｂ＃３、及びＢ＃４が設定されている。これらのバッファに、ポリゴンデータタファイルに記録された闘士のモーションデータが順次記録される。図３６の下欄にはこのことが示されている。ポリゴンデータファイル＃１に記録されているモデル＃１のデータは、残像開始後残像用バッファ＃１に記録され、次の時点では、ポリゴンデータファイル＃２に記録されたモデル＃２のデータは残像用バッファ＃１に記録され、残像用バッファ＃１の内容が残像用バッファ＃２に送られる。すなわち、残像用バッファのデータは、＃１→＃２→＃３→＃４の順序に送られて、残像用バッファ＃１にはポリゴンデータファイルから一つ前のコマにおける実像のモーションデータが送られる。残像用バッファ＃４は、順次そのまま更新される。

【０１３０】

例えば、５フレーム目の時点では、ポリゴンデータファイルにはモデル＃５のモーションデータが格納され、かつこのモデルが実像として表示されている。一方、残像用バッファ＃１にはモデル＃４のモーションデータが記憶され、残像用バッファ＃２にはモデル＃３のモーションデータが記憶され、残像用バッファ＃３にはモデル＃２のモーションデータが記憶され、残像用バッファ＃４にはモデル＃１のモーションデータが記憶されている。

【０１３１】

ポリゴンデータファイルのモーションデータと、残像用バッファのデータ（過去計算されたモーションデータに等しい。）とは同時に表示されるので、画面上には実像と残像とが同時に表示されることになる。図３５はこのことを示している。実像に対しては通常のレンダリングが施されて表示されるのに対して、残像に対しては、演出効果を高めるために、換言すれば「残像らしく見せる」ために、半透明の処理を施しながらレンダリングする。この半透明の程度をより前の時点のコマ数における残層ほど高めることができる。前のフレームの残像ほど透明に表示される。なお、残像用バッファは、ＲＡＭ１０２に設定される。

【０１３２】

次に、この実施例における残像処理の動作について説明する。図３７は、この処理を行う概略フローチャートである。先ず、残像が必要であるか否かが判定される（Ｓ５００）。この判断は、例えば、モーション毎に行われる。例えば、闘士の大きな動きが伴う大技（廻し蹴り、背負い投げ等）には残像が必要として処理される。図３５を参照されたい。その他に、動作量が所定値以上の場合も同様である。

【０１３３】

図３８は、この残像必要性処理の詳細フローチャートである。先ず、技の発生がチェックされる（Ｓ６００）。このチェックは、既述のコントローラ２ｂの操作ボタンや操作スティックの操作状況によって判定される。技が発生したと判定されると、技の属性がデータが読み込まれる（Ｓ６０２）。技の属性データとは、個々の技に対して与えられた性質に係わるデータであり、例えば、「攻撃の技か」、「脚による技か」、「手による技か」等である。

【０１３４】

次に、この属性データが残像を発生すべきデータか或いはその逆かが判定される（Ｓ６０４）。例えば、「攻撃の技」、かつ「脚による技」の場合には、残像表現が必要であるとして、所定のフラグ"Ｆ"を立てて"１"とし（Ｓ６０６）、他の場合には、"Ｆ"を"０"とする（Ｓ６０８）。

【０１３５】

概略フローチャートの次のステップでは、残像必要部位が特定される（Ｓ５０２）。この特定は、詳細フローチャートである図３９に示す処理によって行われる。先ず、Ｓ７００において、攻撃コマンドが発生したか否かがチェックされる。攻撃コマンドとは、操作ボタンや操作スティックの操作が、遊戯者が操る闘士が対戦相手の闘士に対して、攻撃をするような命令態様に至った状態で発生するものである。

【０１３６】

図３９において、例えば、廻し蹴りのコマンドが発生した場合（図３５の場合がこれに該当する。）、攻撃部位が足首のパーツ（図３５の３０Ａ）であることが参照され（Ｓ７０２）、次いで、脚全体の残像を形成するために、足首３０Ａ，下肢３０Ｂ，大腿部３０Ａ、すなわち、左側の脚全体（図３５参照）が残像必要場所として特定される（Ｓ７０４）。なお、ここでの説明のように、闘士は複数のパーツから構成されており、各パーツはそれぞれポリゴンから構成されている。

【０１３７】

もっとも、本実施例の場合のように、攻撃コマンドが発生していない場合にも残像を表示するようにすることができる（Ｓ７０６）。この場合には、当然のことながら、図３７のＳ５００における「残像が必要か？」の判定において、「現在攻撃中」でなくても残像

表現が可能となるように、残像の必要性について判定する。例えば、闘士が技を被って倒れる過程を残像を伴いながら表示するような場合である。この場合は、モーションデータから、Ｓ７０２，Ｓ７０４のように残像が必要なパーツを決定する。

【０１３８】

そして、図３７のＳ５０４では、図３５に既述の残像表示を実行して、Ｓ５０６に至り、残像表示が終了したか否かが判定される。例えば、一連のモーションが終了したか否かが判定され、図３５の例では回し蹴りが終了したか否かが判定され、終了したと判定された場合は、残像表示の処理が終了される。

【０１３９】

この実施の形態によれば、闘士の一連の動きのデータを有するポリゴンファイルのデータをそのまま順次残像用バッファに記録している。このポリゴンファイルには、座標変換されクリッピングされたポリゴン各頂点の座樹を各点での法線ベクトル付きで記録している。このポリゴンファイルのデータをそのまま残像バッファに記録して、これを現在の実像とともに出力するだけであるので、従来技術のように残像用のポリゴンをその都度座標変換（モデリグング）する必要がなく、ＣＰＵブロックに与える計算負荷が軽減される。しかも、パーツ（脚）の過去の位置に残像が表示され、かつ、半透明の演出が加わっているために、残像表現を演出感豊かに表示することが可能となる。なお、半透明の演出に代えて、脚ポリゴンの表示をメッシュ状、あるいはラインによって表現するようにしても良い。半透明の処理によって、脚の残像と背景とが同時に表示される。

【０１４０】

この実施形態によれば、残像が付されるような大技と残像が付されない中・小技とを残像の有無によって、遊戯者は視覚を介して区別できるために、ゲームの演出感（ゲームとしての画像表現の多様化、面白味等）が向上する。

【０１４１】

飛翔体運動処理

次に、飛翔体（設置物）の舞い上げ処理について説明する。この処理は、後述のように、地面の砂、或いは水、また、地面に落ちている葉を舞上げる態様を備える。先ず、前２者のものについて説明する。その概要は次の通りである。

【０１４２】

図３５に示すように、闘士３０が脚を用いて蹴り上げのモーションを採った場合、或いは、闘士が空中より着地した場合、水、砂を舞い上げる処理が行われる。ゲームの演出効果を高めるために、このような処理が実行される。特に、格闘用ゲームのような遊戯者が操作ボタンを激しく操作してキャラクタを迅速に操作しようとするビデオゲームにおいては、キャラクタの動きが多彩、かつ迅速であるため、このような水等の舞い上げをキャラクタの動きに合わせて迅速かつ的確に計算して表現する必要がある。以下に説明する処理がこのことを可能にする。

【０１４３】

先ず、砂、或いは水の跳ね上げ処理について説明する。ここで、跳ね上げ処理とは、図４０に示すように、パターンチェンジするポリゴン（全３２パターン）が計算によって求められた軌道を順次運動することである。このパターンは、水或いは砂が順次拡散する状態をシミュレートしている。砂或いは水の一粒一粒を計算すると、計算時間が掛かるので数粒まとめてパターンチェンジしながら運動させている。

【０１４４】

この処理では、闘士のパーツチェックが行われる。前記闘士は、頭、右手（上腕、下腕、手首）、左手、胸、腹、腰、左脚（大腿部、下肢、足首）、右脚の各パーツから構成されている。パーツチェックは、脚、頭、腰に対して行われる。したがって、例えば、脚により蹴り上げ、頭或いは腰からの落下をシミュレートする。パーツチェックには、パーツの移動量のチェックが含まれる。

【０１４５】

その他、仮想地面の属性フラグ、パーツ位置フラグがチェックされる。仮想地面属性フ

ラグ(b water)が"１（水の属性）"or"０（砂の属性）"、及びパーツ位置フラグ(b air)が"１（着地状態）"or"０（空中状態）"に設定される。(b water)が"１（水の属性）"の場合、水の跳ね上げが行われ、"０（砂の属性）"の場合は砂の跳ね上げが行われる。水又は砂の跳ね上げには、次の態様がある。
着地発生：パーツが地面或いは水面に着地（着水）した場合であり、発生点から砂或いは水が四方に広がる。
蹴り上げ発生：パーツが地面（水面）に着地（着水）している状態で生じ、脚の蹴り上げによって砂、或いは水が脚（チェックしているパーツ）の移動方向に跳ね上げられる。なお、跳ね上げ発生、及び着地発生は、図６０に模式的に示されている。図の２８０は、キャラクタ（闘士）であり、脚を蹴り上げて、砂或いは水２８２を脚の運動特性に応じて跳ね上げている。或いは、着地して水或いは砂を着地動作に応じて、四方に跳ね上げている。

【０１４６】
この処理では、継続処理と称する、着地発生或いは蹴り上げ発生が数インターラプト（数コマ或いは数フレーム）の間継続される処理が実行される。通常、これらの発生時に跳ね上げ数を決定してその１／４をセットする。跳ね上げ数をＮとすると、１インターラプト毎に次のように跳ね上げ数が順次減って行くようになっている。跳ね上げ数をＮとすれば、
１回目　１／４Ｎ，Ｎ'＝Ｎ－（１／４）・Ｎ
２回目　１／４Ｎ'，Ｎ″＝Ｎ'－（１／４）Ｎ'
３回目　１／４Ｎ″
つまり、インターラプト毎に跳ね上げ数の残数の１／４づつセットして行く。跳ね上げの形態は、既述のようにパターンチェンジするようになっている。

【０１４７】
着地発生の場合の処理について説明する。チェックパーツの前回（一つ前のインターラプト）の位置を（0X,0y,0Z）とし、今回の位置を(PX,Py,PZ)とする。地面或いは水面の高さ（Ｙ座標）をeposとする。
パーツの基本速度：
SPDX＝PX-0X　Ｘ方向
SPDy＝0y-Py　Ｙ方向
SPDZ＝PZ-0Z　Ｚ方向
跳ね上げ総数（セット数）：
amount=基本セット数・パーツ移動量
跳ね上げ原点（発生原点）：図４１は、仮想空間の高さ方向（Ｙ方向）より表した模式図である。チェックパーツが矢示の方向に移動しているとき、図４２に示すように、水、砂の跳ね上げ発生原点は、(0X,epos,0Z)のように設定される。

【０１４８】
砂（水滴）の発散処理：チェックパーツの基本速度のＸ，Ｚ成分を抜き出す。図４３は画像生成装置内に仮想的に設定されたモデル空間のＸ－Ｚ平面である。なお、この図はこの平面を斜め上方の位置より見ている状態を示している。このＸ，Ｚ成分のベクトルをＸＺ平面上で＋１３５°乃至－１３５°の範囲でランダムに回転させる。図において、円で囲まれた範囲（ハッチング部分を除く）が砂（又は水滴）が拡散して行く方向である。

【０１４９】
今、ランダムに回転させたベクトルをＡとする。次に、このベクトルをベクトルの方向にＬ移動させてＡ'とする。ここで「Ｌ」は、パーツの半径である。例えば、脚の場合は１０ｃｍ、腰の場合は２０ｃｍ、頭の場合は１５ｃｍに設定されている。このＡ'に基本スピードのＹ成分を加算して最終的に砂、水滴の跳ね上げ速度が決定される。この速度が決定された後、図４０に示す軌跡を採って砂、水滴等がパターンチェンジされながら移動される。よって、図４４に示すように、パーツの回り（半径Ｌ）から砂（水滴）が四方に飛び散る状態がシミュレートされる。

【０１５０】
次に跳ね上げ発生の処理について説明する。図４１に対応する図４５に示す如く、水中或いは砂中にある脚が蹴り上げられるとする。このような動作の場合、チェックパーツの基本スピードを(PX-OX,Py-Oy,PZ-OZ)とする。このベクトルの跳ね上げ角度（COS θ）は、図に示すように、ベクトルの内積を使用して、図４６に示すように計算される。
【０１５１】
ここで、COS θを求める理由は、跳ね上げの角度が大きいほど（９０°）に近い程跳ね上げ量を減らすためである。すなわち、水、砂のセット数は、基本セット数・移動量・COS θとなる。
【０１５２】
砂、水滴のスピードは基本スピードに乱数（０～２５５）／２５５０の数値を乗じた値となる。このことは、先の着地発生の場合においても同様である。乱数を乗じることによって、ランダムに発生する砂、水の飛び散りを多彩に表示することが可能となる。
【０１５３】
このスピードにさらに(OX,Oy,OZ)を加算した点が砂、水の飛び散り発生点となる。図４７を参照すると、乱数が１／１０（＝２５５／２５５０）の場合、発生点は図４７に示す座標によって表示される。なお、砂、水が図４０に示すように順次パターンチェンジ（３２パターン）される。
【０１５４】
既述の継続処理は、このようにして決められた、着地発生点、跳ね上げ発生点を使用して、砂、水の飛び散りが継続的に表示される。
【０１５５】
図４８は、以上説明したこの実施形態（飛翔体運動処理）のフローチャートである。Ｓ１６００において、チェックパーツの下が水面か否かが判定される。この判定は、水或いは砂の跳ね上げが予定されているステージ（シーン）について行われれる。換言すれば、闘士が立っている地面が岩場、土、或いはその他の固形物であるようなステージでは、この判定を行わない。こうすることにより、様々なステージが存在しても、このような飛翔体の処理を必要なステージにのみ迅速に行うことができる。
【０１５６】
パーツの下が地面ではなく水である場合は、次のステップに移行する。このステージでは既述の設置面属性フラグ(b water)が"１"に設定されている。砂上のステージである場合は、砂地面の処理（Ｓ１６０２）に移行する。なお、この処理は、水面上の処理と同じであるので、その説明を省略することとする。
【０１５７】
次の処理、Ｓ１６０４においては、キャラクタ（闘士）の各パーツが着地（着水）しているか否かが、キャラクタの今回ポジションのＹ座標(Py)と水面位置のＹ座標(epos)とを比較することによって判定される。この判定によって、チェックパーツが水面下に在る場合は、着地発生の処理に移行し、これとは反対の判定の場合は、跳ね上げ発生の処理に移行する。
【０１５８】
Ｓ１６０６では、前回のインターラプトにおいて着地状態であったか否かが着地判定フラグをチェックすることによって判定される。このフラグ(b air)が「０」である場合は、数インターラプトの間のこの着地状態が継続される（Ｓ１６０８）。一方、このフラグが「１」である場合は、今回のインターラプトにおいて新たに着地が発生したとして着地発生の処理（Ｓ１６１０）をするとともに、このフラグを「０」にリセット（Ｓ１６１２）して図３４のメインルーチンにリターンする。
【０１５９】
一方、チェックパーツのＹ座標が水面位置より高い場合は、Ｓ１６１４に移行して前回のインターラプトにおいて、着地状態であったか否かが着地判定フラグをチェックすることによって実行される。前回着地状態であったとき、すなわち、チェックパーツが水面下

にあった場合は、例えば、今回図に示すように脚の蹴り上げがあったとして、水を脚の動きに合わせて上方に跳ね上がるパターンが表示される（Ｓ１６１６）。そして、このフラグを「１」にセットして（Ｓ１６１７）リターンする。一方、前回着地状態にないときは、跳ね上げの処理が継続される（Ｓ１６１８）。

【０１６０】

以上説明したように、この実施形態によれば、キャラクタの移動量、移動方向を、キャラクタの動きに合わせて移動する、設置物（水、砂等）に反映させることが可能となる。

【０１６１】

次に、ゲーム中のキャラクタ（闘士）の運動によって風を発生させ、この風によってステージに落ちている葉（飛翔体の一つ）を舞い上げる運動（飛翔体運動処理の一種）をシミュレートとする処理について説明する。

【０１６２】

図４９を参照すると、１７０は運動する闘士を示し、１７２は運動する闘士によって舞い上げられた葉を示す。ここで云うところの「風」とは、数インターラプト（インターラプトとは、一こまの表示のための期間のこと、この場合は１／６０秒、換言すれば、垂直同期信号同士の間）の持続するものではなく、１インターラプト内に葉の挙動に影響を与えるベクトルのことを云う。

【０１６３】

この処理は、図５０に示す概略フローチャートによって実行される。このフローチャートは、キャラクタの運動による風が発生するか否かのチェックを行う（Ｓ１８００，Ｓ１８０２）。次いで、風が発生することをシミュレートする場合は、葉の舞い上がりをチェックするルーチンに移行し（Ｓ１８０４）、次いで、葉の運動をシミュレートする（Ｓ１８０６）。前記ゲーム機は、最初に葉を空中に舞い上げてこれを落下させる状態のシミュレートし、この葉が地面に到達する迄はこの自由落下運動を継続し、この葉が空中から地面に到達した以降は、葉が停止する迄地面上を這う運動をシミュレートする。風が発生しない場合には、葉の舞い上がりチェックを行うことなく、葉の運動を継続する（Ｓ１８０６）。

【０１６４】

先ず、風の発生チェックのためのフローチャートを図５１に基づいて説明する。Ｓ１９００では、パーツポジションを計算する。この計算は、闘士が一対存在する場合には、闘士１，２に対して交互に１インターラプト毎に行われる。チェックするパーツは、右脚，左脚，頭の合計３箇所について行われる。それぞれのパーツの移動量をチェックして風の発生の有無が判定される。この判定は、次のとおりである。

【０１６５】

図に示すように、チェックするパーツの前回（１インターラプト前）のポジションを(0X,0y,0Z)とし、今回のポジションを(PX,Py,PZ)とする。この時の移動量Ｍは、図５１の（１）式に示すとおりである。ここでは、ＸＺ座標系での移動量Ｍ'（式（２））を使用する。

【０１６６】

Ｓ１９０２において、Ｍ'が０．２より大きい場合は、パーツの移動量が風を発生させる程度に大きいとして、次のステップに移行する。Ｍ'がこの値より小さい場合は、風が発生しないものとして、このルーチンを終了する。

【０１６７】

Ｓ１９０４では、Ｙ方向の変化量(Py-0y)が、(Py-0y)≦－０．８である場合、風の向きが極端に下方であるとして風を発生させないものとする。Ｓ１９０６では、風の発生フラグを「風発生あり（＝１）」に立てて、かつ風の発生ポジション、および風の大きさ（ベクトル）を決定する。

【０１６８】

風の発生ポジション及びこのＸ，Ｙ，Ｚ方向のベクトルは、図に示すとおりである。Ｍ'は、０．１２≦Ｍ'≦１．０に修正される。このＭ'を風のＹ方向のベクトルする。Ｘ，

Ｚ方向のベクトルは、Ｘ方向，Ｙ方向の変化量に０．６を乗じた値とする。ここで、０．６を乗じているのは、Ｘ，Ｚ方向のベクトルがＹ方向のベクトルより相対的に小さい値として、主に風がモデル空間の上方に舞い上がる状態をシミュレートするためである。

【０１６９】

風は、一つのインターラプトの間一つベクトルしか発生させないものとする。複数箇所において、風が発生した場合には、その中でもっとも値が大きいものを発生すべき風とする。

【０１７０】

次に、葉の舞い上がりチェックのフローチャートについて説明する（図５２）。葉の一つ一つについて、風発生ポジションと葉のポジションとから葉が風の影響を受けるか否かをチェックして、この影響範囲内であれば、風の影響を受けて葉の運動（スピード，ベクトル）が変化する。

【０１７１】

ステップ２０００では、風のデータと葉のデータが読み込まれる。これらのデータ（Ｘ成分，Ｙ成分，Ｚ成分）は、図に示すとおりである。Ｓ２００２では葉の高さに関する演算を行う。この演算は次のとおりである。地面の高さをeposとすると、地上１ｍの高さはepos＋1.0（ｍ）となる。この高さから葉の高さlyposを引いた値をＣ１とする。

Ｃ１＝epos＋1.0－1 ypos（ｍ）

Ｃ１≦０のときは、葉の位置（地上から１ｍ以上にある。）がキャラクタの動作によって発生した風の影響を受け無い程度に高いと判定されて、この処理を終了する。このとき葉は風の影響を受けない（影響度０％）。一方、風の影響があると判定された時（Ｓ２００４：Ｎｏ）、すなわち、Ｃ１＞０の時は、Ｃ１の値に応じた影響係数を決定する（Ｓ２００６）。例えば、葉が地上にあるとき（1 ypos＝epos，Ｃ 1＝１）は、葉は風の影響の１００％の影響を受ける。Ｃ 1＝０５の時は５０％の影響を受ける。Ｃ１は、０≦Ｃ１≦１．０の値をとる。これらのことは、図６８に説明されている。この影響係数は、葉の高さによって、葉が風の何％の影響を受けるかということを指標するためのものである。

【０１７２】

次のＳ２００８は、風の発生ポジションと葉のＸＺ平面における距離Ｌ'を演算する。図に示すＬ'の計算式中の w yspd は、既述のとおり風のＹ軸方向のベクトルであり、且つ、葉に対する影響距離（半径）も兼ねている。

【０１７３】

w yspd－Ｌ'が０以下のときは、葉は風による影響範囲外として、風による影響は無いものとしてリターンする（Ｓ２０１０）。一方、０を越える場合は、Ｃ２（距離による影響係数）を算出する。さらに、このＣ２を修正してＣ２'を算出する。この影響係数の例が図６８に示されている。

【０１７４】

Ｓ２０１４では、葉に対して風の影響（ベクトル：Ｘ方向，Ｙ方向，Ｚ方向）、すなわち、葉のスピード（ベクトル）を図に示すように決定し、葉の現在ポジションにこの影響を加えて葉の運動をシミュレートする（図５０のＳ１８０６）。葉の運動の処理ルーチンは、このすぐ後に説明される。葉のＸ方向，Ｚ方向のスピードは、常に新しく設定され、Ｙスピードは前回（前のインターラプト）のスピードに加算される。

【０１７５】

次に葉の運動をシミュレートする詳細フローチャートについて説明する（図５３）。葉の運動は、図５４に示すように、葉の中心ポジションの運動と、葉のモデルの X,Yrotの回転運動の二つの形態のものが与えられている。（１）は風の影響や重力によって葉の中心ポジションが移動する運動を示したものであり、（２）は、葉が「ひらひら」と舞う運動をXrot，Yrotによってシミュレート（表現）することを示している。これらの運動は、概略フローチャートで説明した、自由落下運動の一環である。

【０１７６】

前者の運動（空中にある場合、そして空中から着地した場合）の特性式は、図のＳ２１

００に示すとおりである。葉が空中にある場合、Ｘ方向の速度(1 xspd) に０．８７を乗じて新たなＸ方向の速度としている理由は、空気抵抗を考慮に入れたためである。Ｚ方向についても同様である。Ｙ方向の速度は、０．００２を減じて新たなＹ方向の速度としている。これにより、葉の重力方向の速度が徐々に増して行くことがシミュレートされる。但し、Ｙ方向の落下速度(1 yspd)の最大値は－０．００６とする。葉が着地している場合については、後述する。

【０１７７】

一方、図５５に示すように、葉２３０がひらひらと舞う状態は、既述のX rot とY rot とによってシミュレートされる（Ｓ２１０２）。但し、この実施の形態では、３２パターンのX rotを用意して、順次パターンチェンジしている。Y rot による回転は、次のようである。

Ｙ回転速度＝1 yspd×1200×256

１回転(360°)を0×0000～0×FFFF（0～65536）で表現している。１°■182(0 xB6)である。なお、Y rot もパターンチェンジにすると、葉の動きが粗くシミュレートされるおそれがあるので、一方の回転要素（X rot）をパターンチェンジとし、残りの要素（Y rot）を計算によって求めている。

【０１７８】

葉が地面に到達した後の運動は、図５３に示すように、中心運動特性式と回転運動特性式によってシミュレートされる（Ｓ２１００，Ｓ２１０２）。中心運動特性式は、図５３に示されている。この運動特性式では、葉のＹ軸方向の速度（1 yspd）は０に設定されているので、地面上を葉が回転しながら動く状態がシミュレートされる。この特性式において、Ｘ方向の速度とＹ方向の速度のそれぞれに０．５を乗じているのは、地曲の摩擦を考えてのことであり、空中に葉がある場合よりも小さい値を乗じて空気抵抗よりも地面の摩擦抵抗が大きいことをシミュレートしている。尚、Ｘ軸周りの葉の回転角度（X rot）が「０」であるか否かが判定され「０」である場合は、葉が地面に対して平行に接したとして葉の動きを停止させる。したがって、キャラクタが動くことによって発生した風によって葉が空気中に舞った後、この葉が地面に落下して地面と平行になった時点で葉の動きが停止される。

【０１７９】

空中に舞った葉が着地したか否かは、地面のＹ座標と葉のＹ座標を比較して判定される。葉のＹ座標≦（又は「＝」）地面のＹ座標であるときに、葉は着地状態にあると判定する。なお、葉は一つのポリゴンに葉の模様（テクスチャー）を貼り付ける（マッピング）することによって形状モデリングされる。

【０１８０】

以上説明した、飛翔体（水、砂、葉）は、その移動量や移動方向が、キャラクタ（モデル）の動作の特性によって決まるために、モデルの動きを飛翔体に正確に反映させることができる。このような飛翔体は本来モデルの動きによって飛んだり、或いは落下したりするものであり、モデルの動きをこれらに反映させることにより飛翔体の動作を高品質にシミュレートできる。また、以上説明した飛翔体の運動処理によれば、仮想空間内におけるモデル（飛翔体）の（自由）落下運動を演出感豊かに、且つ高品質に再現できる。例えば、葉が「ひらひら」と舞う映像が生成されていることである。

【０１８１】

自由落下運動処理

次に、仮想空間内を落下する対象物の運動処理について説明する。ここでの実施例は、対象物を雪として、雪に「ゆらぎ」運動を与える演出をシミュレートすることについて説明する。ここで説明する「揺らぎ」とは、雪が風によって舞いながら降っている状態、或いは闘士の動きによって舞い上がった落ち葉が舞い落ちる状態等比較的不規則な運動を云う。この実施例は、この「揺らぎ」の状態を効率良くシミュレートするようにしている。

【０１８２】

そこで、雪が舞う実施例について説明する。舞い落ちる雪一つは一つ又は複数のポリゴ

ンによって構成されており、モデル空間に設定されている風のベクトル（このベクトルは適宜変化させている）の影響を受けている。雪のポリゴンの運動は、重力による自由落下、風ベクトルによる影響（以上、請求項の第１の手段）、仮想空間のＸＺ平面（水平面、すなわち地表面に平行な座標系）の円運動（同項の第２の手段）によってシミュレートされる。円運動の角速度、振り幅は、メモリ内にテーブル化されており、その一つが選択されて使用される。

【０１８３】

雪の挙動に揺らぎを与える、既述の円運動は、次のようにして与えられる。角速度をωとし、回転角をｑとし、振り幅をａとすると、揺らぎ運動のＸ成分(ｘ ｏ ff)とＺ成分(ｚ ｏ ff)、すなわち雪の基本挙動（自由落下と風によるベクトルによって決定される。）に対するオフセット量は、次のとおりである。

ｘ　ｏｆｆ＝ｓｉｎ（ｑ＋ω）・ａ

ｙ　ｏｆｆ＝ｃｏｓ（ｑ＋ω）・ａ

このオフセットデータは、毎インターラプトの間１０回計算されて、雪の基本ポジションに加算される。

【０１８４】

雪の運動方程式は、次のとおりである。

ｘ　ｐｏｓ：雪のポジション（Ｘ方向）、

ｙ　ｐｏｓ：雪のポジション（Ｙ方向）、

ｚ　ｐｏｓ：雪のポジション（Ｚ方向）とする。

x pos=x pos＋x off＋wind x（風ベクトルスピードのｘ成分）

y pos=y pos＋fall（雪の落下速度、例－１ｃｍ／ｉｎｔ）

z pos=z pos＋z off＋wind z（風ベクトルスピードのｚ成分）

【０１８５】

本実施例によれば、雪が風に揺られて舞い落ちる挙動（図５６参照）が、前記運動方程式によって表現される。したがって、複雑な雪の「揺らぎ」運動が簡単な計算式によって得ることができ、ＣＰＵブロックに与える計算負荷も大きなものには至らない。

【０１８６】

図５６はこの雪の移動軌跡を示すものであり、所定方向に大きさや方向が変わるベクトルを自由落下する雪に加えても「揺らぎ」の状態を的確にシミュレートすることはできない。そこで、この雪に円運動（周回運動の一種、その他楕円運動のようなものでも良い。）を加えることによって、この「揺らぎ」を再現できるようにしている。

【０１８７】

この処理によれば、仮想空間内のモデルの自由落下運動をシミュレートする際の品質を高くすることができるために、落下画像処理の演出感を豊かに醸し出すことが可能となる。

【０１８８】

不定形区画における衝突判定処理

次に不定形の区画とモデルとの衝突処理について説明する。この処理は次のような内容を有している。ここで、区画とは闘士の動きを規制する領域であり、例えば、周りが不定形の壁で囲まれたリングを想像すると理解し易い。闘士はこのリング内を自由に移動できるが、壁を越えてリング外に移動することはできない。この壁がここで説明する「区画」に相当する。

【０１８９】

図５７の（Ａ）には不定形の区画２５０が示されている。この区画は仮想空間のＸＺ平面（Ｙ軸は高さ方向）に設定されている。この区画は７角形として設定されている。この区画としては様々な形態のものを設定可能である。このような区画としては、周りが岩場であるリングを想定した場合のその岩場を云う。

【０１９０】

図５７はこの処理の原理をも示している。壁（区画）が実際に存在する様に見せるため

には、キャラクタが壁を越えて移動しないよう、キャラクタの位置を補正する必要がある。この実施形態は、大きさや形を任意に設定することができる闘技場の壁とキャラクタとの当たり判定を行い、位置を補正するための壁の法線方向へ追い出しベクトルを算出する。

【０１９１】

図５７の（Ａ）は、この衝突判定のモデル図であり、（Ｂ）はベクトル計算の原理図であり、（Ｃ）はキャラクタ（モデル）と区画との衝突判定モデル図の一部拡大図である。なお、図５７にはベクトル計算式も同時に記載されている。

【０１９２】

（Ａ）に示すように、この当たり判定に際して、キャラクタの形状を球によって近似し、さらに壁及びこの球体を地面と平行な平面（ＸＺ平面）に平行投影して、区画の領域を形成する各辺と、球が投影された結果得られた円２５２との位置関係を調べる。ここで、球の位置はキャラクタの計算上の位置である。この図において、Ｌはこの区画の任意の辺の一頂点であり、Ｒはこの辺の他の頂点である。ある任意の辺について、反時計方向にＬが来るように設定される。

【０１９３】

図中のＴはキャラクタが投影された円の中心点である。ベクトルＲ，ベクトルＴ，ベクトルＬは、この座標系のある任意の一点からこれらの各点に対して設定されたベクトルである。また、ベクトルＰは円の中心座標から区画の各辺に対して垂直に引かれたベクトルである。ベクトルＶはこの円に対する辺ＬＲに平行な接線と円との交点から、辺ＬＲに対して直角に引かれたベクトルであり、この処理における追い出しベクトルに相当する。

【０１９４】

この処理においては、追い出しベクトルＶが算出されることが目的である。円の中心が直線ＲＬの外側（これを正領域とする。）にある場合は、図に示すように、追い出しベクトルは図の式（１）に示されるように設定される。

【０１９５】

一方、円の中心が直線ＲＬの内側（これを負領域とする。）にある場合は、式（２）のように設定される。すなわち、円の中心と直線ＲＬとの間の距離が、半径（ｄ）以上の場合は、追い出しベクトルは設定されない。この距離が越える場合は、追い出しベクトルが設定される。このとき、ベクトルは壁の方向に向かっているために、単位ベクトルＰｅを円の中心が直線ＲＬの外側にある場合とは反対方向に設定される。

【０１９６】

図５８はこの実施例の動作について説明するフローチャートである。先ず、コントローラのボタン或いは方向キーの操作量が読み込まれる（Ｓ２６００）。この操作量から、計算上のキャラクタの位置が決定される（Ｓ２６０２）。すなわち、区画が無いとした場合のキャラクタの計算上の位置であって、コントローラの操作によってこの位置が要求される。

【０１９７】

次いで、Ｓ２６０４において、このキャラクタの位置とある辺ＲＬとの比較が行われる。すなわち、図５７の式（３）の関係がチェックされる。この判定が否定された場合（＞１）は、次の辺ＲＬについてこの関係をチェックする。この場合は、点Ｔからこの辺に直角にベクトルＰを引くことができないとし（すなわち、衝突判定の対象となる辺とはされない。）、Ｓ２６０６のベクトルＰの計算処理をすることなくＳ２６０８に移行する。

【０１９８】

この関係が肯定された場合は、Ｓ２６０６にて図５７に示すところによって、ベクトルＰの値を算出する。次に、Ｓ２６０８にて全ての辺について、Ｓ２６０４の処理が終了したが判定され、否定された場合はＳ２６１０に移行して他の辺についての比較が実行される（Ｓ２６０４）。一方、これが肯定された場合は、Ｓ２６１２に移行し、ベクトルＰの最小値が選択される。このベクトルＰが設定された辺が追い出しベクトルを法線方向に設定するための壁（辺、区画）として決定される。すなわち、このベクトルが設定される辺

の方にキャラクタが接触していると判定される。
【０１９９】
そして、次のステップ（Ｓ２６１４）では、円の中心が区画内にあるか否かが判定され、区画外のときは図の式（１）によって追い出しベクトルが設定され、区画内のときには式（２）によって追い出しベクトルが設定される。式（１）の場合及び（２）において、ベクトルＰの大きさが円の半径より小さい場合は、キャラクタと壁との間に当たり（衝突）があるとされて、既述の追い出しベクトルが設定される。この追い出しベクトルによって、実際の映像では、キャラクタがこの区画（壁）から先に移動することが規制される画像処理が実行される（Ｓ２６１６）。
【０２００】
もっとも、この壁から追い出しベクトルによってモデルが壁から辺の法線方向に定義されたベクトルＶによって弾かれるような画像処理を実行するようにしても良い。図６７はその時に於ける画像処理の一例であり、符号３５０は闘士であり、符号３５２は不定形の壁である。闘士３５２が図中区画の側（図中左方向）に移動しようとしても、区画に遮られて移動できない状態を示している（図の（１）の状態）。但し、図の（２）に示すように、闘士３５０の全体は区画によって移動できない状態であるが、右脚３５４（キャラクタの一部のパーツ）が両方向の矢印に示されるように進退するようにすることはできる。このことにより、遊戯者は自己が操るモデルが区画に遮られて、これ以上その方向に移動できないことをしることができる。
【０２０１】
この実施形態に示される処理によれば、各辺毎に追い出しベクトルを算出するようにしているために、不定形の区画であってもこの区画とキャラクタとの当たり判定を確実に行うことができ（すなわち、追い出しベクトルの値が与えられた時には当たり判定が肯定される。）、この判定結果を画像処理に反映することが可能となる。
【０２０２】
テクスチャー列の作成
次に繰り返すテクスチャー列を作成する形態について説明する。図５９はこのことを示す説明図である。下段（Bottom）には、第１のテクスチャー列が示されており、中段(Top)には第２のテクスチャー列が示されている。上段(Togeter)には、第１のテクスチャー列と第２のテクスチャー列とを重ねた結果の第３のテクスチャー列が示されている。
【０２０３】
第３のテクスチャー列は、所定の周期を持って繰り返す映像「海上の反射パターン」の映像であることを目的に形成されたものである。第３のテクスチャー列は、０～２９の３０カット（枚）の絵から構成されており、これらの絵を順次、ポリゴンにマッピング、すなわち、０～２９番までの絵をマッピングした後、さらに０～２９番までマッピングし、これらを繰り返すことにより、これらのテクスチャー列のつなぎ部分（例えば２８～２番）での絵の変化が自然的（連続的、換言すれば、絵飛びが無い）であるようにしている。これにより、海面上の反射パターンが繰り返す映像を作成できる。
【０２０４】
ここで、第１，２のテクスチャー列は、既述の市販アプリケーションソフトウエアを利用して作成される。テクスチャー列の発生に当たってパラメータの値はそれぞれの列に応じて異なるようにしている。但し、ソフトウエアの制約から、それぞれの列において同じであるが、テクスチャー列の最初の領域と最後領域のテクスチャー群は、つながりの悪い絵柄になっている。すなわち、例えば、第１のテクスチャー列の０番と２９番、そして第２のテクスチャー列の０番と５９番である。
【０２０５】
そこで、この処理においては、第２のテクスチャーに順次変化する透明度のパラメータ（０～１００％）を与える。第２のテクスチャー列の上に記載されている数値がこの透明度に関する値である。１００％とあるのは、透明度が全く無い状態を示し、他に７％とあるのは、透明度がより進んだ状態（透明度９３％）であることを示している。全く透明に

する場合には、０％が与えられる。第１のテクスチャーは、１００％のパラメータを与えている。

【０２０６】

第３のテクスチャー列は、第１のテクスチャー列と第２のテクスチャー列を図示するように順番に重ねられて作成される。すなわち、第１のテクスチャー列の０番と第２のテクスチャー列の３０番が重ねられて第３のテクスチャーの０番が形成され、第１のテクスチャー列の１番と第２のテクスチャー列の３１番が重ねられて第３のテクスチャーの１番が形成され、・・・・、第１のテクスチャー列の２９番と第２のテクスチャー列の５９番が重ねられて第３のテクスチャーの２９番が形成される。

【０２０７】

このとき、第１のテクスチャー列の０番と第２のテクスチャー列の３０番が重ねられるとき、第２のテクスチャー列の３０番の透明度は０であるために（全く透明でない状態）、第１のテクスチャー列の０番は第２のテクスチャー列の３０番に全く隠されて、第３のテクスチャー列の０番の映像は、第２のテクスチャー列の３０番の映像に等しくなる。

【０２０８】

一方、第１のテクスチャー列の２９番と第２のテクスチャー列の５９番が重ねられるとき、第２のテクスチャー列の３０番の透明度は９７％であるために（ほぼ透明に近い状態）、第３のテクスチャー列の２９番の映像は、第１のテクスチャー列の２９番の映像にほぼ等しくなる（換言すれば、第２のテクスチャー列の５９番の映像は殆ど見えない状態）。

【０２０９】

ここで、第２のテクスチャー列を発生させるとき、パラメータの与え方は、その３０番の絵が第１のテクスチャー列の２９番の絵に続くような絵となるようにしているために、第３のテクスチャー列を見た場合には、テクスチャー列のつなぎ部分で（図５９では２８番～２番）において、つながりが既述のように自然となる絵を発生させることができる。

【０２１０】

第１のテクスチャー列の終わり部分（例えば２５番乃至２９番）、第２のテクスチャーの最初の部分（例えば、３０番乃至３４番）の絵がつながるようにそれぞれのテクスチャー列にパラメータを与えることによって（あるいは結果的にこうなることによって）、図の矢示で示すように、透明度も勘案すると、第１のテクスチャー列の２９番から第２のテクスチャー列の３０番につながるようになることにより、第３のテクスチャー列が図示するように自然につながる形態の絵となる。

【０２１１】

第３のテクスチャーは、メモリの所定領域に記憶されており、海面を示すポリゴンに順次マッピングされて、海面の白波（画面上では輝度が高い部分、図２７では白く表示されている部分）の模様や形態が周期的に繰り返す映像を表現することができる。図示する例では、１／３０秒毎に一枚の絵が表示されるとし、３０枚の絵であるとすると、１秒毎に海面の白波部分の模様が周期的に繰り返す映像を再生することができる。

【０２１２】

波の形態を再生する速度、及び使用するテクスチャー数は、適宜変更可能である。画面の中心に闘士が立っているステージがあるとすると、ステージ近傍（すなわち、画面の中心付近）では、既述の再生速度（一コマ当たり１／３０秒、３０コマ再生）とし、これから遠くの波については、コンピュータの計算負荷を低減する観点から、一こま当たりの再生速度を下げ、かつ合計の使用コマ数を少なくすることができる。例えば、第３のテクスチャー列の０番、１０番、２０番，２９番のコマを使用する。このようにしたのは、画面の周辺付近にある映像の再生を多少粗くしても、遊戯者に与える悪印象は少なく、かつ、コンピュータグラフィックス装置の計算負荷を低減する事の方が有利であるからである。

【０２１３】

既述のゲーム装置のメモリには、第３のテクスチャー列が記憶されている。したがって、コンピュータグラフィックスを用いたゲーム機の分野に、既述のキャラクター列を利用

することができる。ここで、説明したキャラクター列の発生方法は、コンピュータグラフィックスの技術分野ばかりでなく、アニメーションの動画作成の技術にも利用することができる。

【０２１４】

スロー再生処理

図６１乃至６４は、スロー再生処理（再生速度変更処理）の原理を説明するためのキャラクタの動作図である。二つの闘士２９０，２９２が互いに向き合って、両者の闘技の優劣を競いあっている状態を示している。図６１において、２９０は攻撃を受けている闘士であり、２９２は攻撃側の闘士である。符号２９４が闘士２９０が闘士２９２から闘技（攻撃：パンチ）を受けた攻撃部位である。図６１と図６１を上方より見た平面図である図６２は闘士２９０が攻撃をそのまま受けている状態を示している。

【０２１５】

一方、図６３とその平面図である図６４に示すように、闘士２９０が闘士２９２からの技を右手のパーツで払い（すなわち、防御）、それが成功すると闘士２９２が矢示の方向に仰け反る（ずれる）ようなモーションが再生される。このとき、闘士２９２のモーションの再生速度を低下させてスロー再生を行う。

【０２１６】

このように、一方のキャラクタ（闘士）が他の闘士に技（「パンチ」）を繰り出したとする。一方、他の闘士がこのパンチを払う技をほぼ同時に出したとする。なお、これらのパンチや或いは払い技は、遊戯者がコントロールパッドの操作ボタンや方向キーを適宜操作することによって発生する。

【０２１７】

このパンチの払いが成立したときに、攻撃側の闘士が後方に大きく仰け反るモーションが再生される。このとき、このモーションの再生速度を防御側の闘士の映像の再生速度よりも小さくなるようにする。すると、このスロー再生により、攻撃側の闘士にいわゆる「隙」が生じて、防御側の闘士から攻撃技を攻撃側の闘士に繰り出し易くなる。闘士２９２は一旦仰け反った後、図６５のように元の体型（仰け反る前の）に徐々に復帰する。

【０２１８】

図６６は、この処理を示すフローチャートである。Ｓ３４００において、防御側の闘士が既述の払い技を出す。Ｓ３４０２では相手側の攻撃技（図の場合は「パンチ」）が払い技に当たったか（すなわち、払い技の動作コマンド下にある手のパーツと攻撃側の闘士の手のパーツの衝突判定）の判定を行う。

【０２１９】

この判定が否定された場合は、スロー再生の処理を行うことなくリターンする。一方、この判定が肯定された場合は、攻撃側のキャラクタが防御側のキャラクタの正面に位置しない場合にはリターンし、正面の場合には、Ｓ３４０６に移行する（Ｓ３４０４）。Ｓ３４０６では、攻撃技が払うことができる技であるか否かを判定する。例えば、パンチであれば払うことができ、キックであれば払うことができない等である。この判定は、技毎の専用フラグを設定してフラグレジスタの内容を参照することにより容易に実現される。

【０２２０】

払い不可能技である場合は処理を終了し、払い可能技である場合はＳ３４０８に移行する。このステップでは、スロー再生されるべきコマ数を設定し、これをタイマーに保存する。次のステップＳ３４１０では、技によって払われる量（図の例では、攻撃側の闘士が仰け反るモーションを再生する場合、各パーツが仰け反る量）を設定する。コマ数及び払い量は、所定の特性式によって計算され、或いはテーブル化されている。

【０２２１】

Ｓ３４１２では払い量を設定されたコマ数で割り、一こま分のリーフ量（空間座標系に於ける移動量、及び移動方向）を算出する。そして、Ｓ３４１４では、攻撃側の闘士の再生速度（表示速度）を防御側の闘士の例えば半分にする。例えば、攻撃側のキャラクタの再生速度を１／６０秒の半分１／３０秒とする。

【０２２２】
次いで、Ｓ３４１６では、前記タイマーの値を判定し、「０」である場合は、スロー再生処理は終了されたとして、攻撃側闘士の再生速度を元の状態に復帰させて（Ｓ３４１７）、リターンする。一方、「０」でない場合は、一こま当たりのリーフ量にタイマーの値を乗じて攻撃側の移動量とする。そして、Ｓ３４２０では、この移動量を攻撃側の闘士の座標値に加算する。すなわち、図５１の攻撃側の闘士２９２の位置に、図６３の払い技が成立した場合の移動量（仰け反り量：本来のポジション（図６１の闘士２９２の位置））を加える。
【０２２３】
次いで、Ｓ３４２２ではタイマーの値（Ｔ）を１減じて、Ｓ３４２４では、攻撃側の闘士２９２（図６３）のモーション位置を再計算することが行われる。そして、メインルーチンにリターンする。 この処理によれば、払い技が成功したときに、攻撃側の闘士が最初に大きく仰け反りながら、スロー再生状態でさらに徐々に仰け反って行く映像が再生される。このとき、防御側の闘士は攻撃技を攻撃側の闘士に対して繰り出し易くなる。
【０２２４】
この処理によれば、一方の闘士のモーションの再生速度を低くしながら他方の闘士のモーションの再生を行うことができるために、遊戯者は自己が操る防御側の闘士技をより有効に繰り出すことができることになり、このことは、ゲームの要素としてのスロー再生を可能にしたことに通づることになる。したがって、スロー再生を演出感豊かに表現することが可能となる。
【０２２５】
なお、本発明は上述した各種の実施例の構成に限定されるものではなく、当業者であれば、請求の範囲記載の構成の範囲内でさらに様々な変形が可能であろう。
【０２２６】
なお、ゲーム機の動作用プログラムが記憶された記憶（記録）媒体としては、既述のカートリッジＲＯＭ、ＣＤ－ＲＯＭの他にインターネット、パソコンネット上の通信媒体でも良く、また、電子メールのサーバーをも含むものである。
【図面の簡単な説明】
【０２２７】
【図１】本発明の実施例に係る画像処理装置としてのビデオゲーム機のブロック図である。
【図２】第１の実施例におけるメインの処理を示すフローチャートである。
【図３】円運動処理におけるキャラクタの円運動を示す斜視図である。
【図４】円運動処理の詳細フローチャートである。
【図５】仮想空間の上から円運動を示した模式図である。
【図６】円運動の終了状態に相当する斜視図である。
【図７】仮想的摩擦処理の原理図である。
【図８】仮想的摩擦処理の詳細フローチャートである。
【図９】投影表示処理の原理図である。
【図１０】交差判定処理の原理図である。
【図１１】交差判定処理の詳細フローチャートである。
【図１２】内接円で近似された構造物のｘｚ平面図である。
【図１３】交差判定処理の具体例を示す原理図である。
【図１４】交差判定処理の他の具体例を示す原理図である。
【図１５】壁状の構造物を近似した態様を示す原理図である。
【図１６】キャラクタのモーションの標準態様を示した斜視図である。
【図１７】キャラクタモーションの補間された態様を示す斜視図である。
【図１８】キャラクタのスプライン関数によって移動する軌跡を示す概念図である。
【図１９】キャラクタのモーション処理制御動作を示すフローチャートである。
【図２０】二つのキャラクタの間に高低差がある状態を示す模式図である。

【図２１】キャラクタの高低差処理を示すフローチャートである。
【図２２】この処理の時間流れを説明するための流れ図である。
【図２３】高低差処理がされた状態を示す模式図である。
【図２４】人物類似のキャラクタの一例の正面を示す、写真である。
【図２５】その第２の例を示す、写真である。
【図２６】コントロールパッドが拡大された正面図である。
【図２７】混合技のファイル構成を示す図である。
【図２８】混合技を展開するためのフローチャートである。
【図２９】混合技処理過程で表示される画面を表示した画面正面図である。
【図３０】同処理過程で表示される他の画面を表示した正面図である。
【図３１】同処理過程で表示される他の画面を表示した正面図である。
【図３２】同処理過程で表示される他の画面を表示した正面図である。
【図３３】同処理過程で表示される他の画面を表示した正面図である。
【図３４】第２の実施例におけるメインの処理を示すフローチャートである。
【図３５】残像処理がされたモデルの正面図である。
【図３６】残像処理の原理図である。
【図３７】残像処理の概略フローチャートである。
【図３８】その詳細フローチャートである。
【図３９】図３７のフローチャートの詳細フローチャートである。
【図４０】跳ね上げられた飛翔体の軌跡を示す図である。
【図４１】飛翔体の着地発生を説明する図である。
【図４２】飛翔体の着地発生を説明する図である。
【図４３】飛翔体の着地発生を説明する図である。
【図４４】飛翔体の着地発生を説明する図である。
【図４５】跳ね上げ発生の説明図である。
【図４６】跳ね上げ発生の際のベクトル線図である。
【図４７】跳ね上げ発生の際のベクトル線図である。
【図４８】飛翔体運動処理のフローチャートである。
【図４９】飛翔体運動処理が適用されたキャラクタの背面図である。
【図５０】この処理の概略フローチャートである。
【図５１】その詳細フローチャートである。
【図５２】その詳細フローチャートである。
【図５３】その詳細フローチャートである。
【図５４】飛翔体の運動形態を示す図である。
【図５５】飛翔体の運動形態を示す図である。
【図５６】自由落下運動処理がされた対象物の移動軌跡を示す図である。
【図５７】不定形区画とモデルとの衝突判定処理の原理図である。
【図５８】そのフローチャートである。
【図５９】テクスチャー列の作成原理図である。
【図６０】モデルによって飛翔体の跳ね上げ及び着地発生が生じた状態を示す正面図である。
【図６１】スロー再生処理の原理を説明するためのキャラクタの動作を説明するための側面図である。
【図６２】図６１を上方から見た平面図である。
【図６３】スロー再生状態にあるキャラクタの動作を説明するための側面図である。
【図６４】図６３を上方から見た平面図である。
【図６５】本来の位置に復帰した状態のキャラクタを示す側面図である。
【図６６】この処理の動作フローチャートである。
【図６７】不定形区画とモデルとの衝突判定処理によって表現されるモデルのモーション図である。

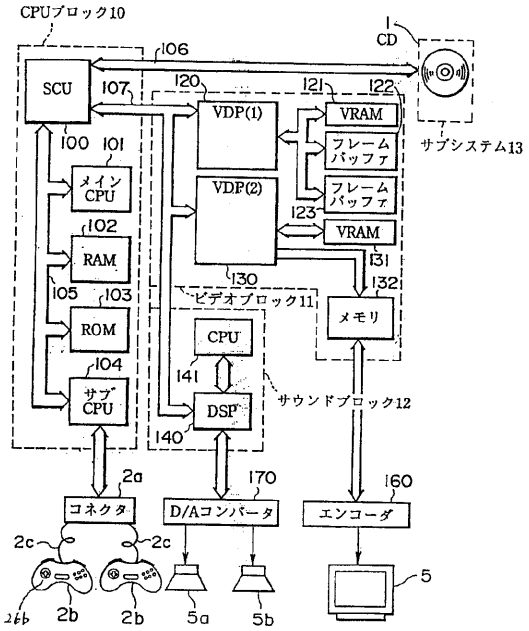
【図６８】図５４の飛翔体運動処理における葉が風によって葉の高さ毎に影響を受ける係数を説明する図である。

【符号の説明】

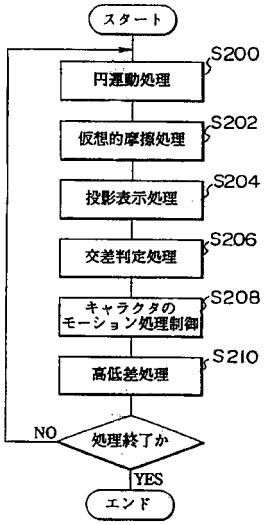
【０２２８】

１……ビデオゲーム機、２ｂ……コントロールパッド、５……モニタ装置、１０……ＣＰＵブロック、１１……ビデオブロック、１２……サウンドブロック、１３……サブシステム、３０、３２、１６０、１７０、２００、２０２、２９０、２９２、３５０……闘士、１７２、２３０……葉

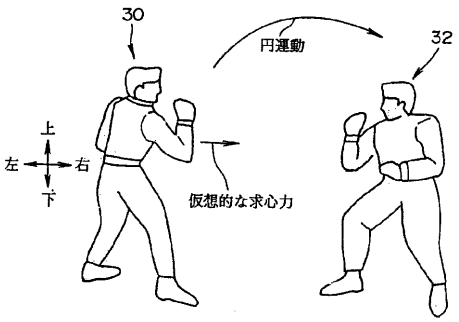
【図１】



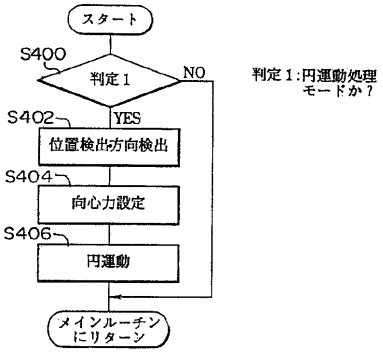
【図２】



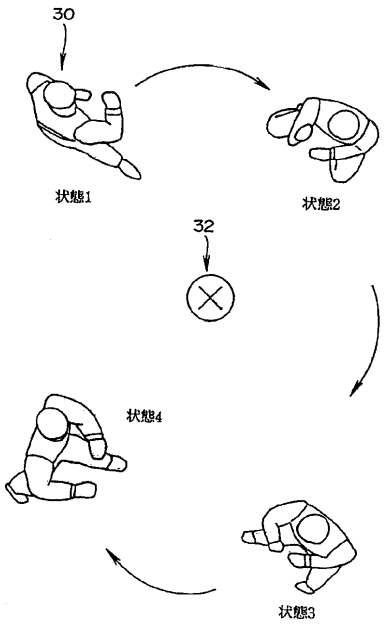
【図３】


30
円運動
32
上
左
右
下
仮想的な求心力

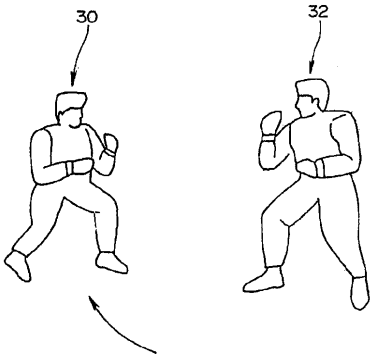
【図４】


スタート
S400
判定１
NO
判定１:円運動処理
モードか？
S402
YES
位置検出方向検出
S404
向心力設定
S406
円運動
メインルーチン
にリターン

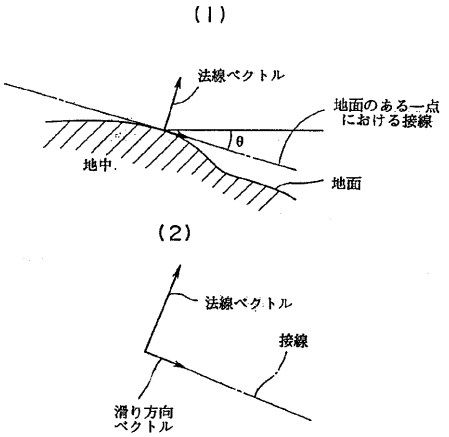
【図５】


30
状態1
状態2
32
状態4
状態3

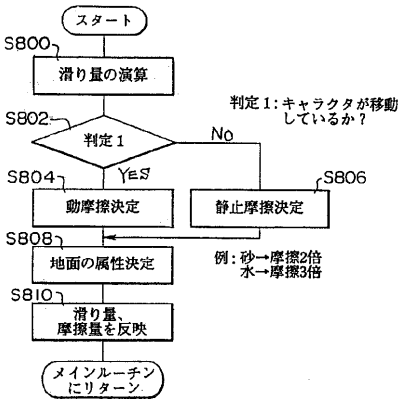
【図６】


30
32

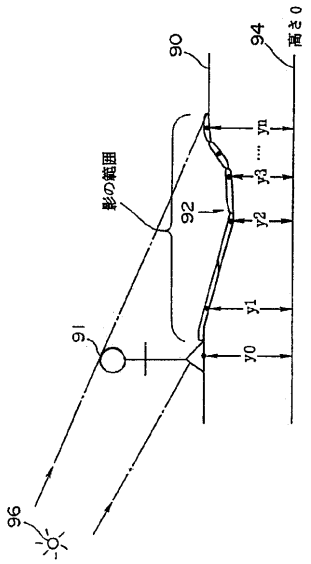
【図７】


(1)
法線ベクトル
地面のある一点における接線
θ
地中
地面
(2)
法線ベクトル
接線
滑り方向ベクトル

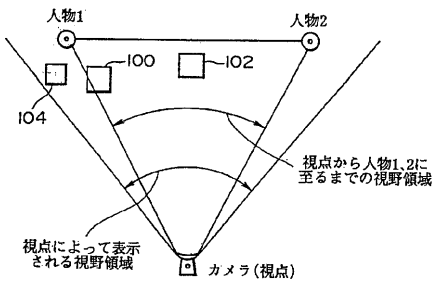
【図８】


スタート
S800
滑り量の演算
判定1：キャラクタが移動しているか？
S802
判定1
No
YES
S804
動摩擦決定
S806
静止摩擦決定
S808
地面の属性決定
例：砂→摩擦2倍
水→摩擦3倍
S810
滑り量、摩擦量を反映
メインルーチンにリターン

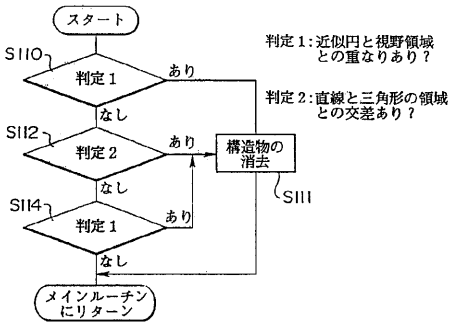
【図９】


90
94
高さ0
ym
y3
y2
y1
y0
影の範囲
92
91
96

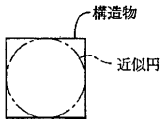
【図１０】


人物1
人物2
100
102
104
視点から人物1、2に至るまでの視野領域
視点によって表示される視野領域
カメラ（視点）

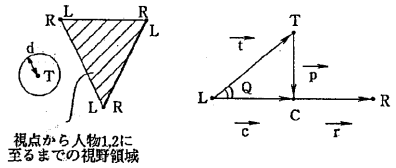
【図１１】



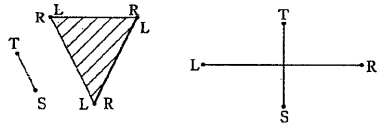
【図１２】



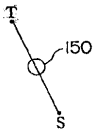
【図１３】



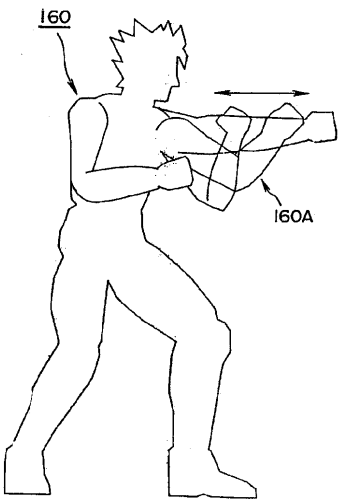
【図１４】



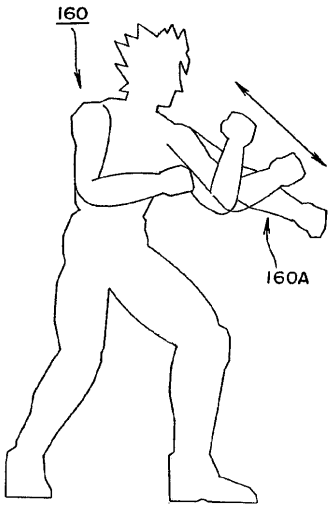
【図１５】



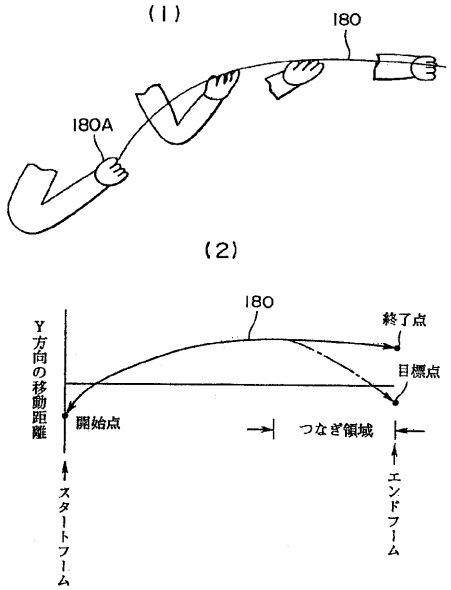
【図１６】



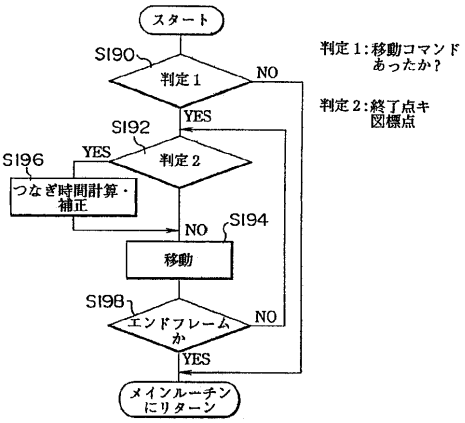
【図１７】



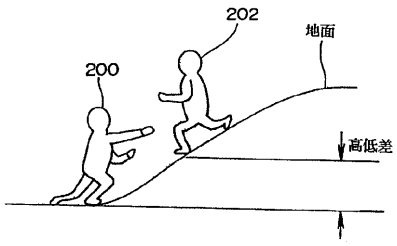
【図１８】



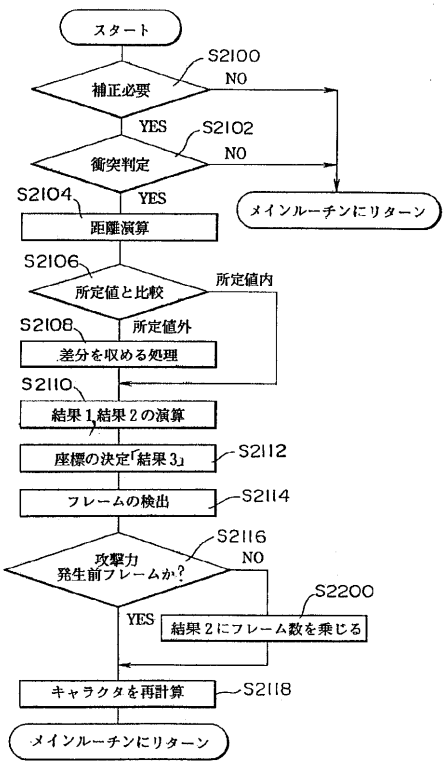
【図１９】



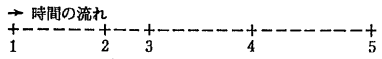
【図２０】



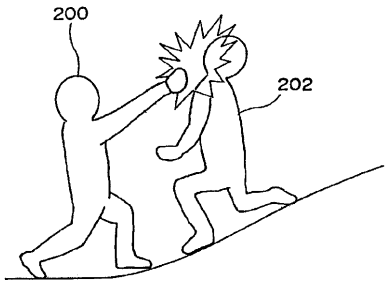
【図２１】



【図２２】



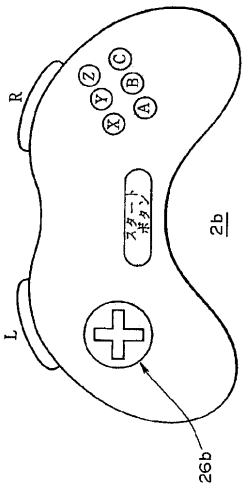
【図２３】



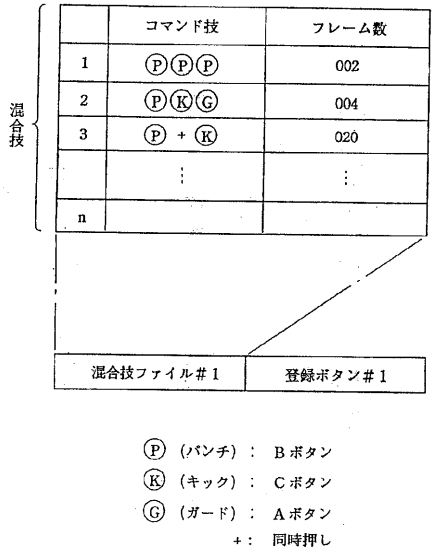
【図２４】

【図２５】

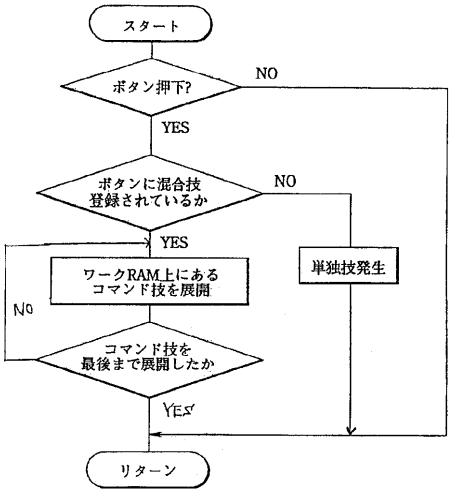
【図２６】



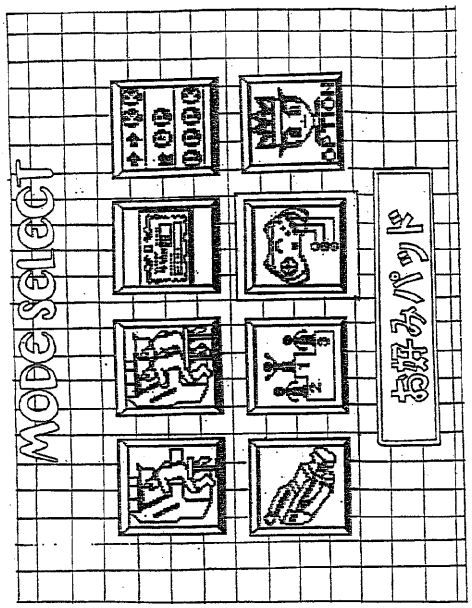
【図２７】



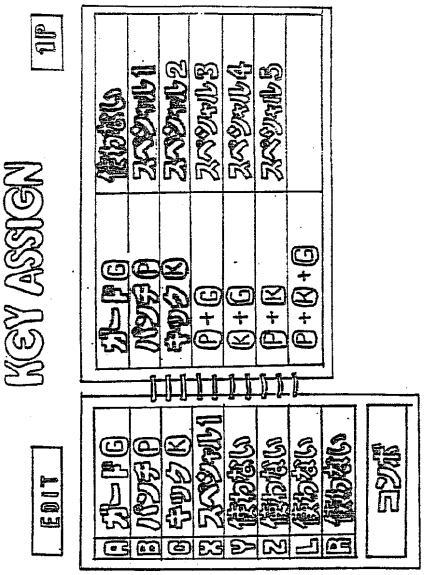
【図２８】



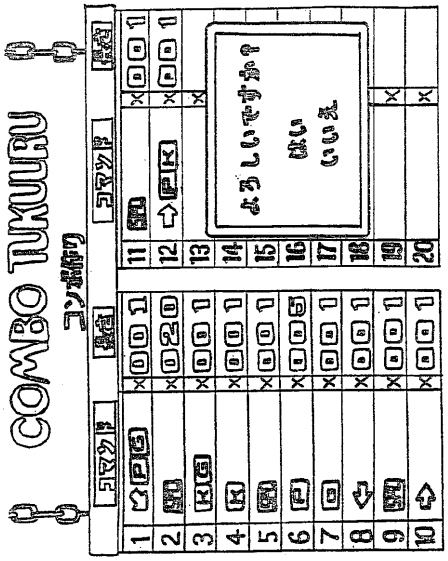
【図２９】



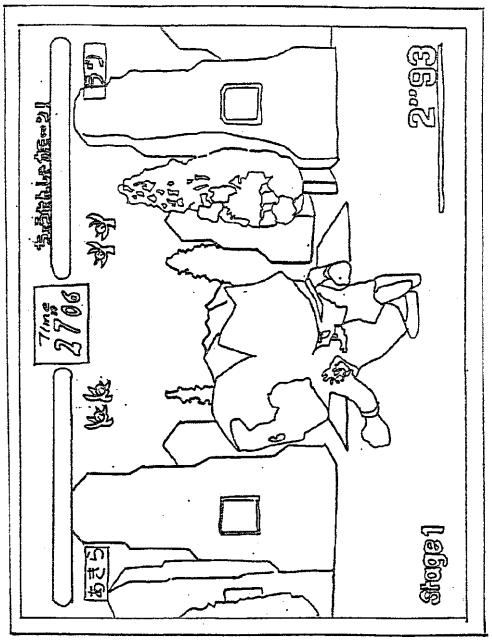
【図３０】



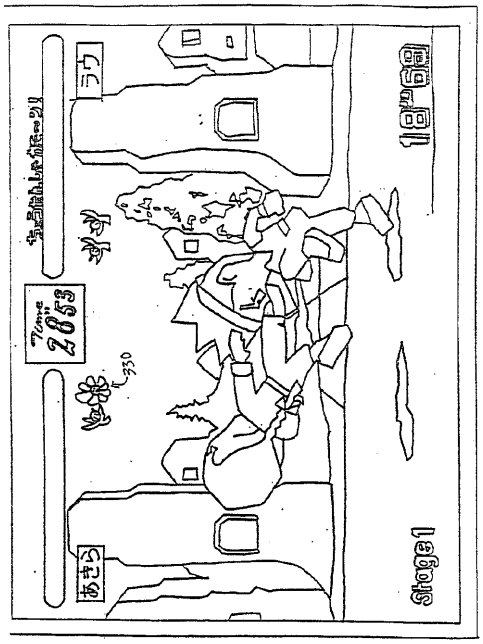
【図３１】



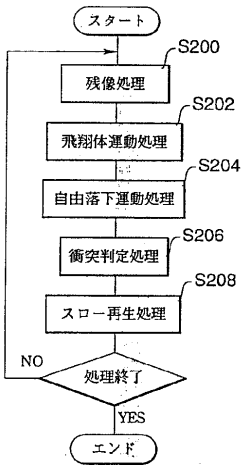
【図３２】



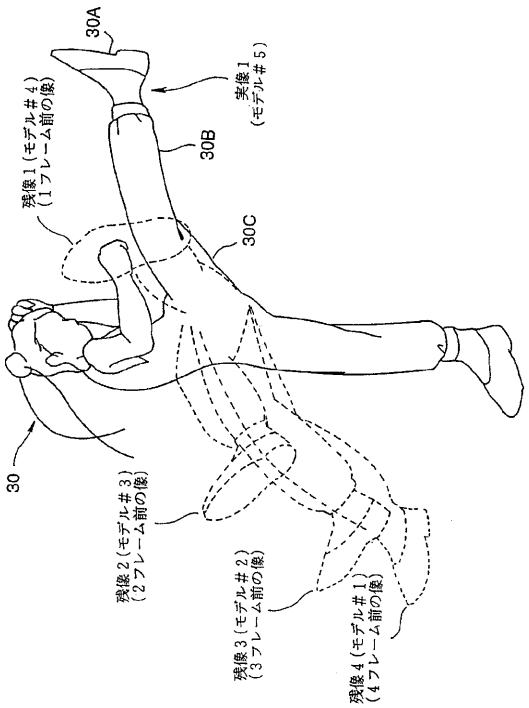
【図３３】



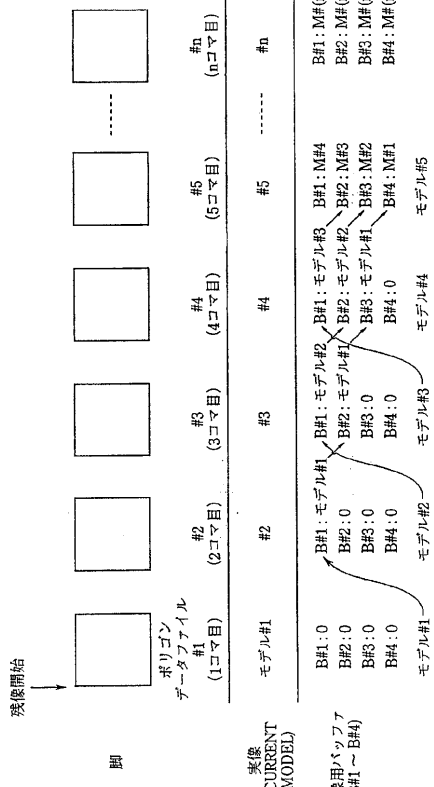
【図３４】



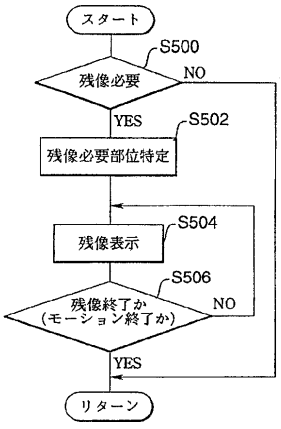
【図３５】



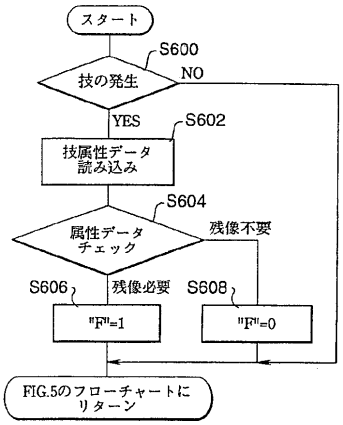
【図３６】



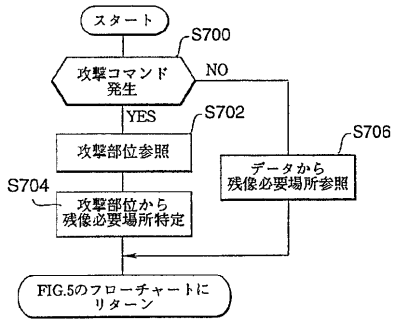
【図３７】



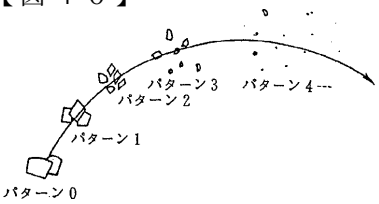
【図３８】



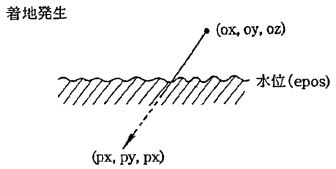
【図３９】



【図４０】

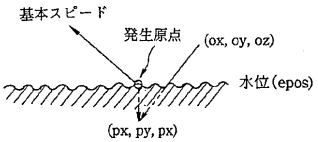


【図４１】

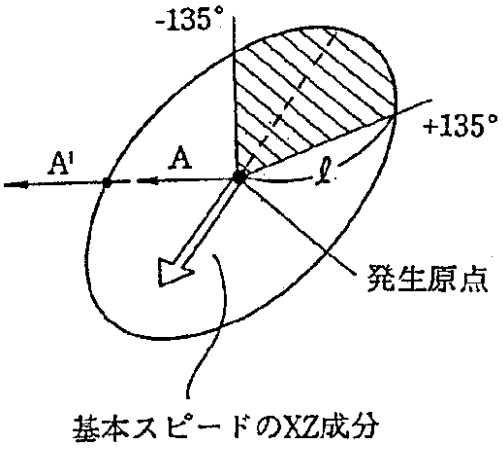


【図４４】

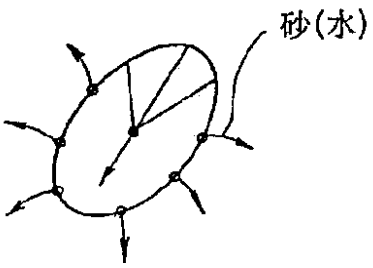
【図４２】



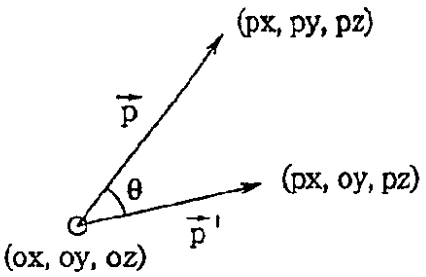
【図４３】



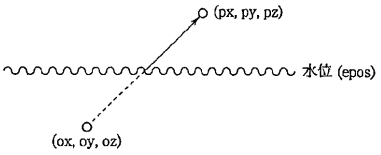
【図４６】



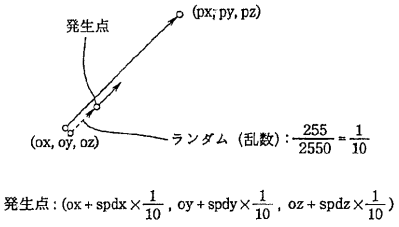
【図４５】



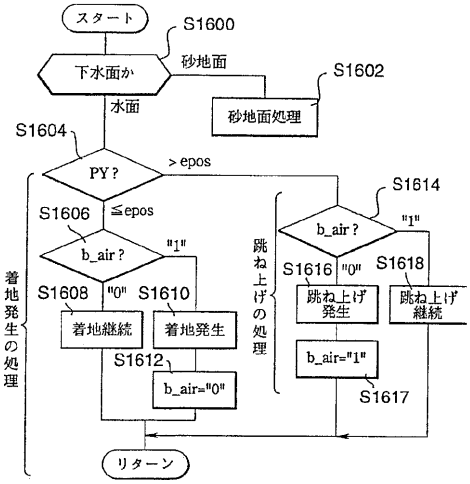
$$\cos\theta = \frac{\vec{p} \cdot \vec{p}'}{|\vec{p}| \cdot |\vec{p}'|}$$



【図４７】

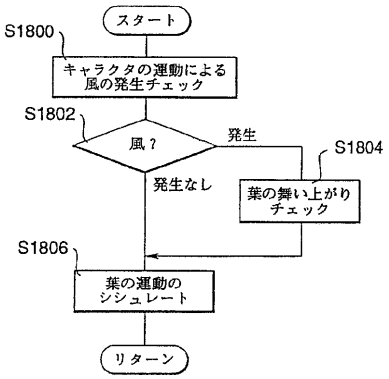


【図４８】

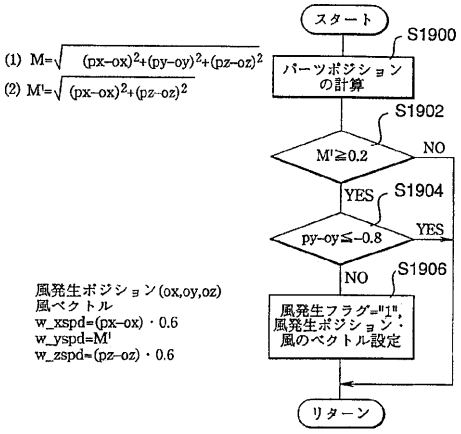


【図４９】

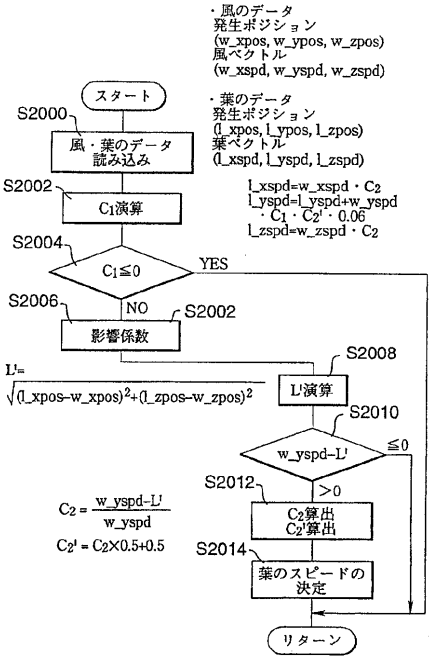
【図５０】



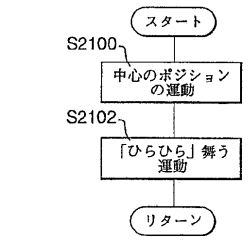
【図５１】



【図５２】



【図５３】



S2100

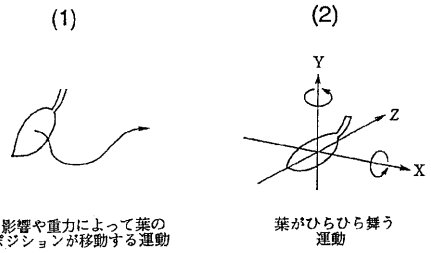
空中の場合

l_xspd = l_xspd×0.87
l_yspd = l_yspd− 0.02
l_zspd = l_zspd×0.87

着地している場合

l_xspd = l_xspd×0.50
l_yspd = 0
l_zspd = l_yspd×0.50

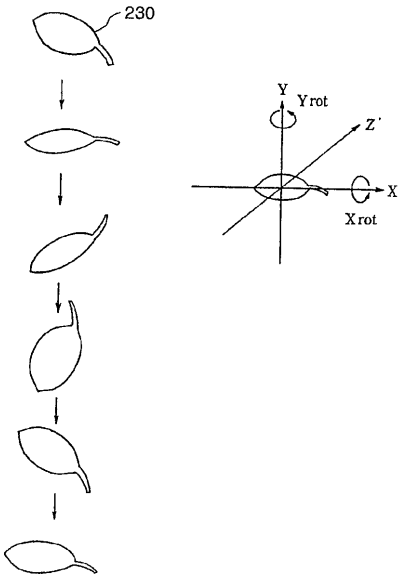
【図５４】



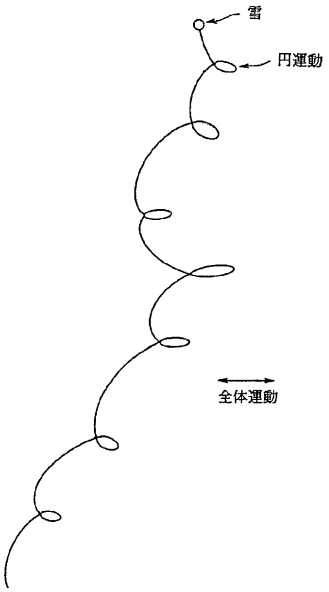
風の影響や重力によって葉の中心ポジションが移動する運動

葉がひらひら舞う運動

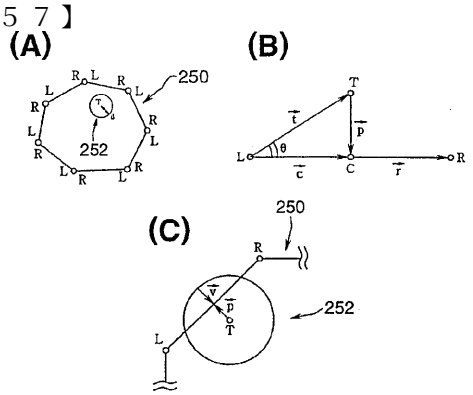
【図５５】



【図５６】



【図５７】



$\vec{r} = \vec{R} - \vec{L}$

$\vec{t} = \vec{T} - \vec{L}$

$\vec{p} = \vec{c} + \vec{L} - \vec{T}$

$cx = rx \cdot \frac{|\vec{c}|}{|\vec{r}|}$ ， $cz = rz \cdot \frac{|\vec{c}|}{|\vec{r}|}$

ここで

$|\vec{c}| = |\vec{t}| \cos\theta$

内積の定理より

$|\vec{r}| \cdot |\vec{t}| \cos\theta = rx\,tx + rz\,tz$

$$\frac{|\vec{c}|}{|\vec{r}|} = \frac{rx\,tx + rz\,tz}{|\vec{r}|^2} = \frac{rx\,tx + rz\,tz}{rx^2 + rz^2}$$

ただし

$$0 < \frac{|\vec{c}|}{|\vec{r}|} < 1 \quad \text{——} \quad (3)$$

追い出しベクトルを $\vec{v}$, $\vec{p}$ の単位ベクトルを $\vec{pe}$ とすると、
点Tが直線LRの正領域にある場合

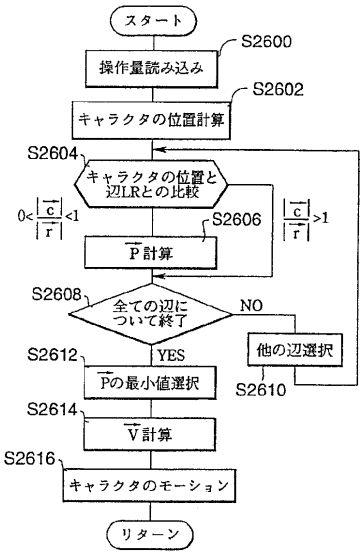
$\vec{v} = (|\vec{p}| + d) \cdot \vec{pe}$ —— (1)

点Tが直線LRの負領域にある場合

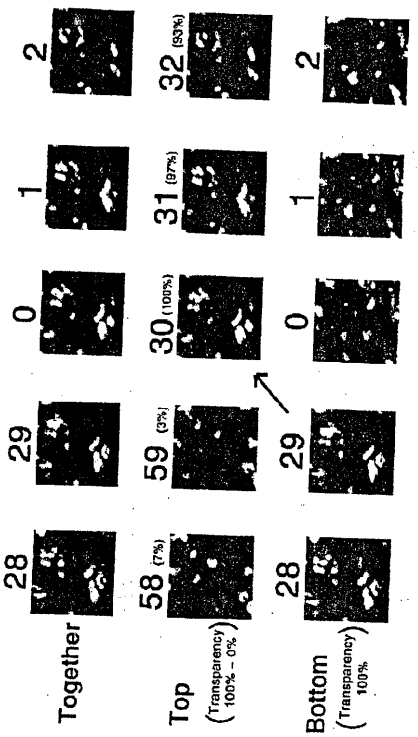
$|\vec{p}| - d < 0$ ならば $\vec{v} = (|\vec{p}| - d) \cdot (-\vec{pe})$

$|\vec{p}| - d \geq 0$ ならば $\vec{v} = \vec{0}$ } (2)

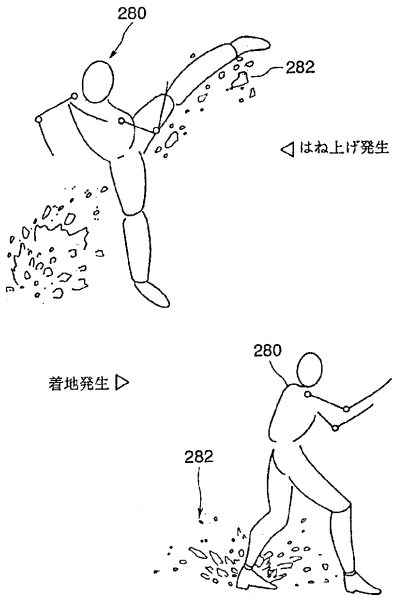
【図５８】



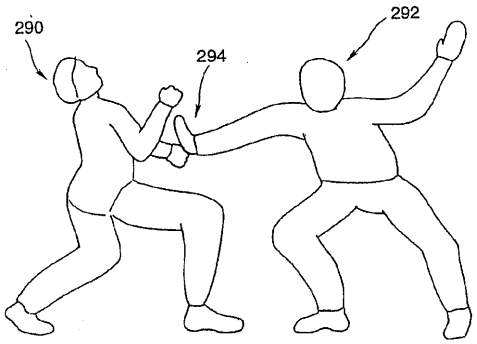
【図５９】



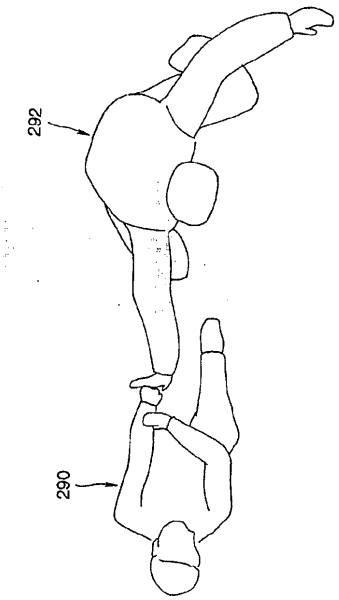
【図６０】



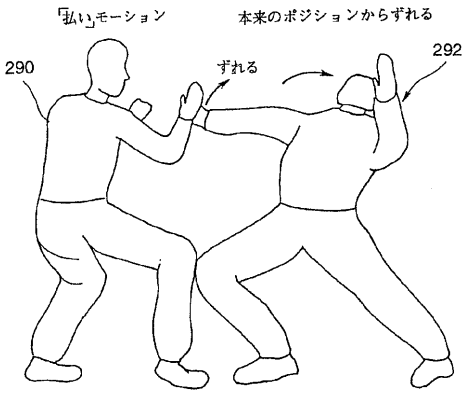
【図６１】



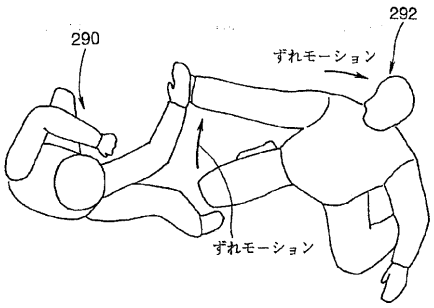
【図６２】



【図６３】

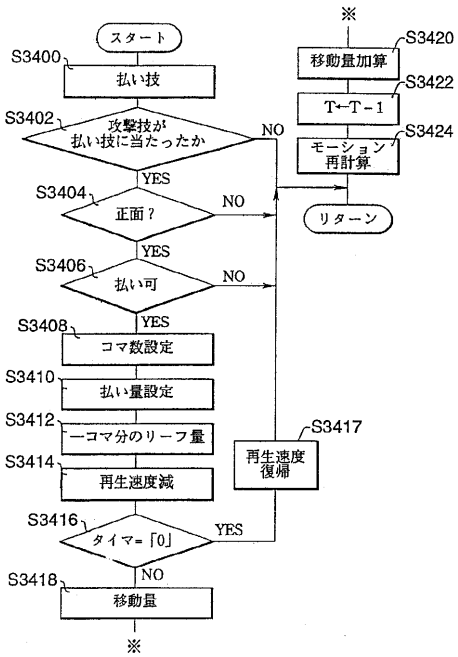


【図６４】

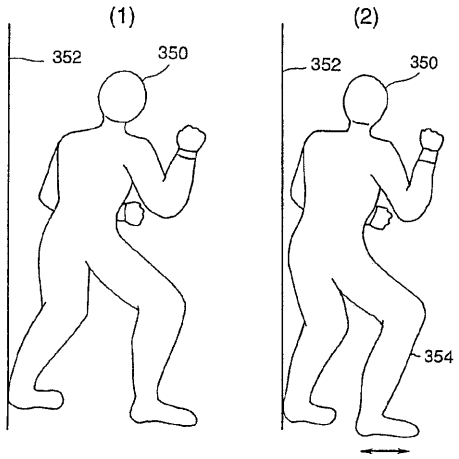


【図６５】

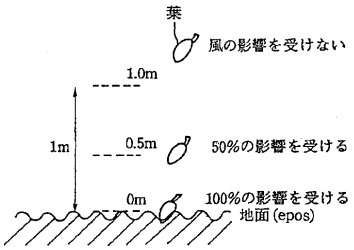
【図６６】



【図６７】


(1)
352
350
(2)
352
350
354

【図６８】


葉
風の影響を受けない
1.0m
1m
0.5m
50%の影響を受ける
0m
100%の影響を受ける
地面 (epos)

フロントページの続き

(72)発明者　小野　孝志
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　芳賀　憲夫
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　大崎　誠
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　山口　貴之
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　関根　紀裕
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　油井　亮弥
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　西川　さおり
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　杉本　哲也
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　吉田　茂
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　中谷　学
東京都大田区羽田１丁目２番１２号　株式会社セガ内
(72)発明者　内田　真澄
東京都大田区羽田１丁目２番１２号　株式会社セガ内

Ｆターム(参考)　2C001 AA06　BA02　BC00　BC01　BC03　BC05　CA01　CB01　CB02　CB04
CB05　CC02　CC08　DA04
5B050 BA08　BA12　EA24　EA26　FA02