(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第3734048号
(P3734048)

(45)発行日　平成18年1月11日(2006.1.11)　(24)登録日　平成17年10月28日(2005.10.28)

(51)Int.Cl.		F I		
G06T 11/80	(2006.01)	G06T	11/80	E
A63F 13/00	(2006.01)	A63F	13/00	B
G06F 3/14	(2006.01)	G06F	3/14	310B
G06F 9/06	(2006.01)	G06F	9/06	410T
G09G 5/377	(2006.01)	G06F	9/06	530P

請求項の数 12　(全 52 頁)　最終頁に続く

(21)出願番号　特願平7-284251
(22)出願日　平成7年10月31日(1995.10.31)
(65)公開番号　特開平8-241427
(43)公開日　平成8年9月17日(1996.9.17)
審査請求日　平成14年4月10日(2002.4.10)
(31)優先権主張番号　08/332813
(32)優先日　平成6年10月31日(1994.10.31)
(33)優先権主張国　米国(US)

(73)特許権者　000233778
任天堂株式会社
京都府京都市南区上鳥羽鉾立町１１番地１
(74)代理人　100098291
弁理士　小笠原　史朗
(72)発明者　大和　聡
京都府東山区福稲上高松町６０番地　任天堂株式会社内
(72)発明者　西海　聡
京都府東山区福稲上高松町６０番地　任天堂株式会社内
(72)発明者　鈴木　利明
京都府東山区福稲上高松町６０番地　任天堂株式会社内

最終頁に続く

(54)【発明の名称】ビデオグラフィックプログラム編集システムおよび方法ならびにそれに用いられる重ね合わせ制御回路

(57)【特許請求の範囲】

【請求項１】

ビデオグラフィックプログラムを編集するための対話型計算システムであって、

ビデオグラフィックプログラムを実行する操作とビデオグラフィック表示出力を生成する操作とが可能である第１の処理装置と、

ビデオグラフィックプログラム編集操作を遂行する操作と前記第１の処理装置が前記ビデオグラフィック表示出力を生成した際に前記第１の処理装置のメモリに保持されたデータに基づいて編集関連表示出力を生成する操作とが可能である第２の処理装置と、

表示画面に表示するために前記編集関連表示出力を前記ビデオグラフィックプログラム表示出力に重ね合わせる重ね合わせ制御回路とを備える、ビデオグラフィックプログラム編集システム。

【請求項２】

前記第２の処理装置は、前記表示画面に表示するキャラクタに応じて前記第１の処理装置のメモリに保持されている表示位置の座標についてのデータに基づいて前記編集関連表示出力を生成する、請求項１に記載のビデオグラフィックプログラム編集システム。

【請求項３】

前記表示画面にビデオグラフィック表示を生成するために、前記第１の処理装置と前記第２の処理装置とに結合された少なくとも１つのピクチャ処理ユニットをさらに含む、請求項１に記載のビデオグラフィックプログラム編集システム。

【請求項４】

前記第２の処理装置に結合された制御レジスタと、前記第１の処理装置に結合される第１のメモリと、使用時に第１の処理装置に結合される第２のメモリを含む着脱可能なカートリッジとをさらに含み、前記第１の処理装置は、前記制御レジスタの状態によって、最初に前記第１のメモリからの命令を実行し、次に第２のメモリからの命令を実行する操作が可能である、請求項１に記載のビデオグラフィックプログラム編集システム。

【請求項５】

ビデオグラフィックプログラムを実行する操作とビデオグラフィック表示出力信号を生成する操作とが可能である第１の処理装置と、ビデオグラフィック編集操作を遂行する操作と前記第１の処理装置が前記ビデオグラフィック表示出力信号を生成した際に前記第１の処理装置のメモリに保持されたデータに基づいて編集関連表示出力信号を生成する操作とが可能である第２の処理装置とを有するビデオグラフィックプログラムを編集するための対話型計算システムにおいて、

オーバレイ制御出力信号を生成するための少なくとも１つのオーバレイ制御入力信号を受信するオーバレイ制御信号生成回路と、

前記ビデオグラフィックプログラム表示出力と編集関連表示出力とを受信するために結合され、前記オーバレイ制御出力信号の状態によって前記表示出力信号の１つを選択的に出力するためのゲート回路とを備える、重ね合わせ制御回路。

【請求項６】

前記オーバレイ制御出力信号は、前記第１の処理装置および前記第２の処理装置の少なくとも１つからの透過制御関連信号である、請求項５に記載の重ね合わせ制御回路。

【請求項７】

前記オーバレイ制御信号生成回路は、前記第１の処理装置の表示出力と前記第２の処理装置の表示出力との透過優先度を制御する少なくとも１つの入力信号を制御する、請求項５に記載の重ね合わせ制御回路。

【請求項８】

表示画面を有する表示装置を含む対話型計算システムを用いてビデオグラフィックプログラムに関連した表示フレームの特有のシーケンスを作成する方法であって、

修正される表示フレームのシーケンスを有するビデオグラフィックプログラムを実行するステップと、

所望する編集ポイントにおいて表示フレームのシーケンスを停止して、編集すべき表示フレームを表示するステップと、

編集ウインドウを前記表示フレーム上に重ね合わせ、それによってユーザが複数の編集オプションの中から選択することができるステップとを備える、ビデオグラフィックプログラム編集方法。

【請求項９】

修正される表示キャラクタのユーザ識別を検出するステップをさらに含む、請求項８に記載のビデオグラフィックプログラム編集方法。

【請求項１０】

ユーザが識別したキャラクタに関連して遂行する編集操作を選択するための編集ウインドウを表示するステップをさらに含む、請求項９に記載のビデオグラフィックプログラム編集方法。

【請求項１１】

前記キャラクタのキャラクタパターンを変更するステップをさらに含む、請求項１０に記載のビデオグラフィックプログラム編集方法。

【請求項１２】

ユーザが識別したキャラクタに関連した移動パターンを変更するステップをさらに含む、請求項１０に記載のビデオグラフィックプログラム編集方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、概略的には独自のビデオグラフィックコンピュータプログラムを生成するための方法および装置に関する。より特定的には、本発明は、コンピュータプログラムまたはビデオゲーム作成方法をよく知らないユーザのために主に設計されたビデオゲーム作成システムに関する。そのようなユーザは、編集を開始した時点から残りのゲームプレイ中持続する編集変更を伴って、ビデオゲームを実行、停止、編集および再開することができるアイコン駆動型対話型計算システムによって、独自のビデオゲームを容易に作成することができる。

【０００２】

【従来の技術および発明が解決しようとする課題】

市場で入手可能なビデオゲームシステムでは、ユーザは、ゲーム中の種々の時点において、以降のゲームプレイを制御する幅広い範囲のゲームプレイオプションを選択することができる。例えば、ユーザは、表示キャラクタの移動を制御して、種々の代替可能な方法で表示画面を出ることができる。ユーザの出口の選択の仕方によって、表示キャラクタは異なる世界に入る。そのようなユーザが選択したゲームプレイングオプションは、元のゲームプログラムの一部であり、どのような方法によっても変更されない。

【０００３】

先行技術では、ユーザが、ビデオゲームプログラムが動作する所望の方法を、限られた事項について修正することができるような初歩的な試みがなされていた。この製品では、ゲーム変更装置は、従来のビデオゲームカートリッジに物理的に挿入され、その結果、ビデオゲームカートリッジは、マイクロプロセッサを基本としたビデオゲームコンソールに結合される。ゲーム変更装置は、ゲームの進行中に変更されるであろう読み出し専用メモリ（ＲＯＭ）記憶コードを含む。本装置は、ビデオゲームマイクロプロセッサのアドレスとデータバスとを監視し、期待値と一致すれば、マイクロプロセッサに対して代替コードを送信する。代替コードは、キャラクタの生命の数や発射可能なミサイルの数等のようなゲームプレイの特徴を修正する。この製品では、ユーザは、ゲーム編集処理に対する制御を行うことができず、本発明に従った現実的に実現可能な方法のようには、ゲームプレイを大幅に変更することができない。さらに、ゲームプレイを変更することができても、可能な動作の範囲内でゲームプログラマが元来想定していた動作およびグラフィック表示を可能にするにすぎない。

【０００４】

ビデオゲーム設計の専門家は、独自のゲームの設計を支援するために、ゲームプログラム作成ツールをこれまで利用してきた。そのようなプログラミング作成システムでは、非常に単純な事項であっても、開発中のゲームを修正するためには、プログラム設計者による非常に多くの作業がしばしば必要とされる。例えば、ゲーム中のキャラクタに対して行われる変更は、通常まず、キャラクタ画像を作成した創作者が規定した元のキャラクタ配列内において行われる。キャラクタに対するあらゆる変更は、新規のファイルとして保管され、例えば開発中のゲームプログラムに変更を導入するプログラムデバッグモジュールへ送信されなければならない。開発中のゲームのグラフィックの変更は、たとえ比較的単純な修正に関してであっても、修正の再コンパイル、再格納および表示という複雑な処理を通常含む。幅広い種々の複雑な変更が、従来の作成プログラムの制御下で作成中のゲームプログラムに対して行われうるが、そのような変更には、高度なプログラミングおよびゲームプログラミング技術に関する知識が要求される。

【０００５】

【課題を解決するための手段および発明の効果】

本発明に従えば、比較的技術の低い小学生から技術の高いゲーム開発者までのユーザが、独自のビデオゲームを簡単に作成することができる。独自のハードウェアおよびソフトウェアプラットホームによって、ユーザは、より詳細なエディタ画面をアクセスするアイコンを選択することによって独自のゲームを作成して、移動オブジェクトおよびゲームの背景に関する幅広い種々のゲーム表示の特徴を直接変更することができる。

【０００６】

所望するゲームのジャンルからのモデルゲームを含むモデルソフトウェアはビデオゲームＲＡＭカセットに、オペレーティングシステムソフトウェアは、システムＲＡＭに、（おのおの）フロッピーディスクを介して格納される。本発明によって、ユーザは、モデルゲームプレイを開始することと、所望するあらゆるゲーム画面においてゲームを停止して「システムブレイク」編集セッションを開始することとができ、編集セッションでは、幅広い種々の編集機能に対する制御を可能にするためのシステムウインドウが、ゲーム画面に重ね合わせられる。その後、ユーザは、修正を行うために、移動オブジェクトまたは背景場面を選択する。例えば、ユーザが移動オブジェクトを選択する場合は、選択された移動オブジェクトは、ユニット番号によって識別され、このユニット番号は、幅広い範囲のゲームプレイ関連の特徴に関連している。移動オブジェクトが選択されると、さらにアイコンが表示されて、ユーザは、オブジェクトに関連する１つまたはそれ以上のポーズについてのオブジェクトのキャラクタドットパターン、オブジェクトに関連する動画特徴、オブジェクトに関連して検出されたゲームプレイ条件に関連する応答、各オブジェクトに関連するステータスの集合、オブジェクトの移動パターン、オブジェクトの移動パターンに関連する音声および幅広い種々のその他のゲームプレイ関連の特徴を、完全に変更することができる。画面背景も、同様に、ステージウインドウをアクセスすることにより修正可能で、背景マップ全体、背景に関連する音楽および幅広い種々のその他の背景関連の特徴を編集することができる。

【０００７】

本発明の実施例は、多重プロセッサを基本とするゲームプロセッサコンソールを用いており、このコンソールは、編集処理およびオペレーティングシステムタスク実行を制御するメイン中央処理装置（ＣＰＵ）と、着脱可能なＲＡＭカートリッジ内に格納されるモデルビデオゲームを実行するためのゲームＣＰＵとを含む。モデルビデオゲームは、ユーザがモデルゲームの所望する面を用いて独自のビデオゲームを容易に作成することができる開始ポイントを提供する。モデルビデオゲームは、全く新しいゲームのように見えるほど、容易に修正されうる。本システムによって、ユーザは、ゲームのあらゆる移動オブジェクト、背景画面、音楽または効果音を修正することができる。

【０００８】

メインＣＰＵおよびゲームＣＰＵは、ゲーム実行および編集処理の際には協働して、メインＣＰＵによって生成された編集画面が、プログラム実行ＣＰＵによって生成されたゲーム画面上に重ね合わされる。ゲーム処理コンソールは、幅広い種々の周辺装置に結合されるポートを含み、周辺装置には、標準的なテレビ受像機、キーボード、ゲームハンド操作器、マウス、モデムボード、ゲーム処理装置をパーソナルコンピュータシステムに結合するためのインタフェイスボード、フロッピーディスクドライブ、外部ＲＡＭゲームカートリッジおよびユーザの識別カードが含まれる。

【０００９】

本システムは、特有の「ユニット」を基本とするデータ構造を用いる。データ構造内では、移動オブジェクトがユニットを基本として処理され、各オブジェクトにはユニット識別が割り当てられる。ユニット識別は、幅広い範囲のオブジェクト、ゲームの特徴、ゲーム処理および位置データに関連し、これには、ステータス情報、現在の画面表示位置、オブジェクトフォーマット、キャラクタサイズ、ポーズ情報、衝突しきい値情報、テンポ情報、属性データ、動画データに加えて、識別されたオブジェクトに関連する他の処理関連情報を識別するアドレスポインタが含まれる。幅広い範囲の情報が、同様に、本明細書中で「ステージ」データと呼ばれる、背景画面関連のデータ構造内に記憶される。プログラミングは、条件および処理関連の操作テーブルおよびユニットポインタを用いて、ユーザが編集しやすいように構築され、操作テーブルおよびユニットポインタは、処理されるオブジェクトユニットデータ構造を識別する。ユニット操作テーブルは、ユニットを処理するために用いられ、所定の条件と、各ユニットの所定の条件を検出して遂行される処理操作とを識別する。条件および処理操作は、ユーザがこれらのテーブルを修正することにより、変更され得る。

【００１０】
本発明のこれらおよび他の目的、特徴、局面および利点は、以下の図面とともに考慮されれば、本発明の実施例に関する以下の詳細な説明によって、より明確となるであろう。
【００１１】
【発明の実施の形態】
図１は、本発明の典型的な実施例に従った、ビデオグラフィックおよびビデオゲームプログラムを作成するシステムの一般的なブロック図であり、ゲームプロセッサシステムユニットコンソール２および多くの関連入出力装置を示す。ゲームプロセッサシステムユニットコンソール２は、例えば３．５インチフロッピーディスク８を受けるための（図２においてより詳細に示す）フロッピーディスク接続ポート７を含む。フロッピーディスク８は、モデルゲームおよびオペレーティングシステムソフトウェアを含む「モデルソフトウェア」を記憶する。オペレーティングシステムソフトウェアは、後でより詳細に説明するように、ゲームプロセッサシステムの全操作を制御し、この制御には、ゲームプログラム関連情報を、ゲームプログラムを実行するＲＡＭカセット４内のメモリへ転送するのを制御することが含まれる。フロッピーディスク８上のモデルソフトウェアは、基本ゲームプログラムソフトウェアを含み、この基本ゲームプログラムソフトウェアは、ユーザのゲーム作成を支援して、ユーザが独自のビデオゲーム設計を行う際には修正される。基本ゲームプログラムソフトウェアは、生成可能なビデオゲームの基本ジャンル、例えば「射撃ゲーム」、「ロールプレイゲーム」、「教育ゲーム」、「シミュレーションゲーム」などを定義する。フロッピーディスク８は、システムが適切な構成であるかどうかを決定するためにブートアップ時に検査されるシステム構成データをさらに含んでもよい。
【００１２】
プログラム情報をフロッピーディスク８からＲＡＭカセット４内のプログラムＲＡＭへ最初に転送した後にはいつでも、ＲＡＭカセット４はゲームプロセッサシステムユニットコンソール２から取り出されて、例えば本出願人がスーパー・ニンテンドウ・エンターテイメント・システムとして販売しているビデオゲームシステムなどの従来のビデオゲームシステムと共に用いられることができる。図２４に関連して後述するように、ＲＡＭカセット４は、また、米国特許番号第４７９９６３５号（対応；特開昭６１－２９６４３３号公報）に示す型の機密保護プロセッサを含む。
【００１３】
ゲームプロセッサシステムコンソールユニット２は、ゲームプロセッサ識別カード６用の挿入ポート５を含む。本発明の典型的な実施例において、識別カードは、フロッピーディスク８上の所定の位置に記憶されているデータと比較される機密保護コードを含む。その比較の結果、権限を有することが判断されると、フロッピーディスク８からのデータは、ＲＡＭカセット４へ転送される。フロッピーディスク８の内容は複写することができるが、ユーザには１枚の識別カード６しか発行されないことによって、偽造者に対する機密保護手段が提供される。識別カード６は、また、ユーザの写真および／または他の識別データを含んでもよい。
【００１４】
図１に示すゲームプロセッサシステムコンソールユニット２は、幅広い種類の入出力装置に結合されるように設計されており、ゲームプレイ中に使用する音声信号の入力を可能にするマイクロフォン１０の接続用のジャックを含む。さらに、ゲームプロセッサシステムコンソール２は、米国特許番号第５２０７４２６号（対応；実開平４－４２０２９号公報）に記載している型のような少なくとも２人プレイヤ操作器１２および１４を含む。これらの操作器は、例えば上述の特許に記載の市販のＳＮＥＳ操作器において一般的に利用されている方法により、移動オブジェクトキャラクタの動作および関連する特殊効果を制御することによって、ユーザのテレビ画面上でビデオゲームプレイを制御するために用いられる。
【００１５】
ゲームプロセッサシステムコンソール２は、さらに、従来のマウス１６とＰＣコンパチブ

ルキーボード１８を接続するためのポートとを含む。マウス１６およびキーボード１８は、後に詳細に説明するような方法により、ビデオゲーム設計処理中にグラフィック／ユーザインタフェイスとしてユーザによって用いられる。マウス１６は、例えば小学生などのゲームプログラミング技術に全く不慣れなユーザが、本件に記載のアイコン駆動型システムを使用して特有のビデオゲームを作成することを可能にする。キーボード１８は、例えば、ゲームプログラマなどのより高度な技術を身につけたユーザが、ゲームソフトウェア命令を入力することによってゲーム修正ができるようにする。

【００１６】

ゲームプロセッサシステムユニットコンソール２は、また、モデムボード２２用の接続ポートを含む。モデム２２は、単なる例として、９６００ボーの半二重モデムである。モデム２２は、ゲームプロセッサシステムを後述のオンラインネットワークで使用できるようにする。システムは、また、システムに電力を供給するための交流アダプタ２０を含む。

【００１７】

ゲームプロセッサシステムユニットコンソールは、後にまた説明するように、２個の中央処理ユニットを含む。１つは主にビデオゲームプレイを制御する役割を果たし、もう１つは主にゲーム編集関連タスクと、ゲーム処理ディスク８からＲＡＭカセット４へ情報を転送するのを制御するためのオペレーティングシステムプログラムを実行するという役割を果たす。

【００１８】

図１に示す上述のシステム構成要素は、ビデオゲームプログラム開発に全く不慣れなユーザが、フロッピーディスク８に記憶されているモデルソフトウェアを用いて、広範囲のビデオゲームを作成することを可能にする。図１に示すオプションの構成要素を用いることによって、システムは、ゲームプログラム設計の専門家が特有のエミュレーションシステムでビデオゲームをより容易に作成できるように拡張されてもよい。この代替実施例においては、ゲーム処理システムは、入出力および他の構成要素を、例えばコンソール２内のオペレーティングシステムＣＰＵにさらに結合させる拡張ボード２４を含む。図１に示すように、様々な追加の入出力構成要素は、スキャナ３０、ハードディスクドライブ２８、またはプリンタ２６のようなシステムに結合されてもよい。スキャナ３０は、従来の光学的スキャナであってもよく、グラフィック画像を走査したり、ビデオゲームを設計する際に用いるゲーム処理システムユニットコンソールメモリシステムに記憶させるためにその画像をデジタル化するのに用いられる。その後、ユーザは、記憶された画像にアクセスしたり、色を付けたり、画像を修正することができる。ＳＣＳＩインタフェイスは、ＩＢＭコンパチブルＰＣ２７に結合することができるように、拡張ボード２４に実施されてもよい。

【００１９】

図２は、ゲームプロセッサシステムコンソールユニット２の斜視図である。コンソール２は、電源オンオフスイッチ１１およびリセットボタン９を含む。リセットスイッチ９は、オペレーティングシステム実行ＣＰＵおよびゲームＣＰＵを含む全システムのリセットを行うことができる。リセットボタン９は、ゲームプログラムを実行するＣＰＵを周知の初期状態にするのに加えて、オペレーティングシステムＣＰＵを中断させて、例えば、遂行するべき操作をテストするのに役立つ。図１および図２に示すように、ソケット５および７は、それぞれゲームプロセッサ識別カード６およびフロッピーディスク８を挿入するためのスロットである。ソケット５および７は、ユーザが識別カード６またはフロッピーディスク８を容易に掴んで取り出すことができるような関連埋込形領域を有する。図２に示すように、コンソールユニットは、また、フロッピーディスク取り出しボタン３を含む。さらに、図１および図２に示すように、コネクタ１３、１７、１９、２１および２３は、それぞれ、マイクロフォン１０、キーボード１８、マウス１６、操作器１２および操作器１４をすぐに接続できるように露出されている。

【００２０】

図３および図４は、図１に示すゲームプロセッサシステムのより詳細なブロック図であり

、例えばコネクタピンの数、メモリ装置の特殊な型などの詳細を規定する説明的な実施例である。これらおよび他の実施態様の詳細は、単に、本発明を実施するための可能な多くの代替組み合わせの内の１つの説明的な組み合わせとして説明されるものであり、本発明の有効範囲を限定するものとして解釈されるべきではない。

【００２１】

図３および図４は、図１に示すのと同一の多数の入出力装置に同じ符号を付けたものであり、図１よりもより詳細に入出力装置とゲームプロセッサシステムコンソール２間の接続を示す。操作器１２、１４、キーボード１８およびマウス１６は、図４のフロントボード１９７に実施されるフロントボードコネクタユニットを介して、ゲームプロセッサシステムコンソール２に結合される。続いて、コネクタは、コンソール２上の２５ピンコネクタ１９１に結合される。マイクロフォン１０は、同様に、マイクロフォンジャックと、フロントボード２５ピンコネクタに同様に結合されるマイクロフォン増幅器とを介して、コンソール２５のピンコネクタ１９１に結合される。マイクロフォン増幅器は、フロントボード１９７上の音声制御によって制御される。

【００２２】

識別カード６は、識別カード回路ボード１８５を介してコンソール２に結合され、識別カード回路ボード１８５は、識別カードコネクタと８ピンコネクタとを有し、８ピンコネクタは、続いて、８ピンコネクタ１９０においてコンソール２に接続される。ＲＡＭカートリッジ４は、８０ピンコネクタ１８９において、６２ピンコネクタボード１８６を介してコンソールに結合され、６２ピンコネクタボード１８６は、ゲームカートリッジ４と接続するための６２ピンコネクタと、コネクタ１８９と接続するための８０ピンコネクタとを含む。ゲームプロセッサ入出力システムは、また、ＬＥＤディスプレイおよび３ピンコネクタを有するＬＥＤボード１８７を含み、３ピンコネクタは、続いて、コンソールユニット２上の３ピンコネクタ１８８に接続される。リセットスイッチ９は、ゲームＣＰＵ２００のリセットを行うために結合される。モデムボード２２は、モデムスロット１９５を介してコンソールユニット２に結合される。ＳＣＳＩボード２４は、拡張スロット１９４を介してコンソールユニット２に結合される。図１に示すフロッピーディスク８を受け入れるフロッピーディスクドライブ１９９は、３４ピンコネクタ１９３を介してコンソールユニット２に結合され、３４ピンコネクタ１９３は、フロッピーディスク操作器２４０に結合される。

【００２３】

図４は、交流アダプタ２０と電源ジャックとを介して交流電力を受けるインタフェイスボード１９８を含み、当該電源ジャックは、ボード１９８上の３０ピンカードエッジコネクタに結合され、当該３０ピンカードエッジコネクタは、コンソールの３０ピンカードエッジコネクタ１９２に結合される。インタフェイスボード１９８は、また、電源ジャックと３０ピンカードエッジコネクタとに結合された直流－直流変換器を含む。ボード１９８の３０ピンカードエッジコネクタは、コンソールユニット２からビデオ信号、同期信号および音声信号を受信し、そのような信号を標準テレビユニット１５に入力する。

【００２４】

次に、ゲームプロセッサシステムコンソール２の内部の２個のＣＰＵアーキテクチャにおいて、図３および図４からわかるように、ハードウェアは、後述するように、ゲームプログラムを実行するＣＰＵ２００（ＳＣＰＵ）および関連システム構成要素と、オペレーティングシステムおよび編集システムであるメインＣＰＵ２２８および関連システム構成要素とを含む。さらに、コンソールユニット２は、プロセッサ間通信の大部分が行われるゲートアレイ回路２２６と、ゲームプログラムを実行するサブシステム出力に編集およびオペレーティングシステムのビデオ出力を重ね合わせるための関連重ね合わせ操作器２１６とを含む。

【００２５】

まず、ゲームプログラムを実行するサブシステムについて説明すると、好ましい実施例において、ゲームプログラムを実行するハードウェアは、ニンテンドウ・オブ・アメリカが

スーパー・ニンテンドウ・エンターテイメント・システム（ＳＮＥＳ）として現在販売しているハードウェアによって実施されてもよい。しかしながら、本発明は、スーパーＮＥＳ関連ゲームプログラムを実行するシステムに限定されておらず、むしろ代替のゲームハードウェア装置で使用されてもよい。ゲームプログラムを実行するゲームＣＰＵ２００は、例えば６５８１６コンパチブルマイクロプロセッサであってもよい。ゲームＣＰＵ２００は、例えば１メガビットの記憶装置を含む作業用ＲＡＭ２０２に結合される。ゲームＣＰＵ２００は、システムデータバス、アドレスバスおよび制御バスを介して、ピクチャ処理ユニット（ＳＰＰＵ）２１２に結合され、ＰＰＵ２１２は、例えば５１２キロビットの記憶装置を含むビデオＲＡＭ２１４に結合される。

【００２６】

典型的な実施例においては、２個のピクチャ処理ユニット（ＰＰＵ）２１２および２２４を使用する。ＰＰＵ２１２は、ゲームＣＰＵ２００の制御下でビデオゲーム表示を生成するための回路を含み、ＰＰＵ２２４は、オペレーティングシステムＣＰＵ２２８の制御下で編集関連ビデオ表示を生成する。ゲームＣＰＵ２００は、垂直および水平ブランキング期間中に、ＰＰＵ２１２を介してビデオＲＡＭ２１４にアクセスする。このように、ゲームＣＰＵ２００は、ライン走査期間以外であってＰＰＵ２１２がビデオＲＡＭ２１４にアクセスしている時にのみ、ＰＰＵ２１２を介してビデオＲＡＭ２１４にアクセスすることができる。ＰＰＵ２２４は、ビデオＲＡＭ２２０からビデオ表示を生成する。図４に示すように、ＰＰＵ２１２の出力およびオペレーティングシステムＰＰＵ２２４の出力は、重ね合わせ操作器２１６を介してユーザのテレビ１５に入力され、また、重ね合わせ操作器２１６は、既に識別されたテレビ１５へのコネクタを介して、ＲＧＢエンコーダ２１８へ結合される。

【００２７】

ゲームＣＰＵ２００は、また、関連作業用ＲＡＭ２１０に結合される音声プロセッサ２０８に結合される。音声プロセッサ２０８は、実行されているビデオゲームプログラムに関連する音声を生成するための市販の音声チップを含んでもよい。ゲームＣＰＵ２００は、音声プロセッサ２０８を介してのみ作業用ＲＡＭ２１０にアクセスすることができる。

【００２８】

ＳＮＥＳシステム内で見い出される構成要素に加えて、典型的な実施例は、１メガビットモニタＲＯＭ２０４と、２５６ビットモニタＲＡＭ２０６とをさらに含む。モニタＲＯＭはプログラム命令を含んでおり、プログラム命令はゲームＣＰＵ２００によって実行されて、その結果、編集および情報転送機能を遂行するための制御をオペレーティングシステムＣＰＵ２２８へ転送する。モニタＲＡＭ２０６は、そのような処理に関連するデータを記憶する。

【００２９】

スーパーＮＥＳビデオゲームマシンについては、その一部が図３および図４においてブロック図で示されているが、本件では一般的な説明しかなされていない。（オペレーティングシステムプロセッサによって用いられる）ＰＰＵ２１２およびＰＰＵ２２４を含むＳＮＥＳに関するさらなる詳細は、本件で明示的に援用している１９９４年７月５日に発行された米国特許第５３２７１５８号（対応；特開平３－６３６９５号公報）の「ビデオ処理装置」に記載されている。スーパーＮＥＳに関するさらなる詳細は、１９９４年３月１日に発行された米国特許第５２９１１８９号（対応；特開平４－１８１２９１号公報）の「直接メモリアクセス装置ならびにそこで用いられる画像処理システムおよび外部記憶装置」に記載されている。

【００３０】

ゲームプロセッサシステムコンソール２は、また、オペレーティングシステムまたは例えば３２ビットＲＩＳＣプロセッサであるＮＥＣＶ８１０プロセッサのようなメインＣＰＵ２２８を含む。代わりに、例えばモトローラ６８０００シリーズプロセッサまたは同様の処理能力を有する他のプロセッサを用いてもよい。後で詳述するように、メインＣＰＵ２２８は、オペレーティングシステム関連ルーチンを、ビデオゲームプログラムを実行する

ゲームＣＰＵ２００と同時に処理する。メインＣＰＵ２２８およびゲームＣＰＵ２００は、図２８に関連して詳細に説明するゲートアレイ２２６を介して通信する。上述したように、（ゲームプロセッサ２００と同様に）プロセッサ２２８は、グラフィック処理操作を遂行するための関連するＰＰＵ２２４を含み、ＣＰＵ２２８にかかるグラフィック処理の負担を軽減する。オペレーティングシステムＣＰＵ２２８は、バスセレクタ２２２の制御設定に応じて、直接にまたはＰＰＵ２２４を介して間接的に、ビデオＲＡＭ２２０に結合される。上記で述べたように、ＰＰＵの２１２および２２４は、本件に援用している上述のＳＮＥＳ関連特許に記載の型である。

【００３１】

本発明の好ましい実施例に従うと、メインＣＰＵ２２８は、３２ビットバス幅を有するプロセッサであり、２１．４７７ＭＨｚのクロック速度で動作する。ＣＰＵ２２８は、例えば４メガバイトＤＲＡＭ２３０に結合される。作業用ＤＲＡＭ２３０は、所望するならば、拡張ＤＲＡＭ２３２を介して２４メガバイトまで拡張可能である。メインＣＰＵ２２４は、初期プログラムローダ（ＩＰＬ）サブルーチンと、ＢＩＯＳオペレーティングシステムプログラムと、ＣＰＵ２２８によってアクセスされるキャラクタフォントとを記憶する、例えば８メガビットＲＯＭ２３４にさらに結合される。ＣＰＵ２２８は、また、システムバスを介して、例えばフロッピーディスクドライブ１９９からのデータをソフトウェアバックアップするために用いられる８メガビットフラッシュメモリ２３６にアクセスする。メインＣＰＵ２２８システムバスは、また、例えば１０ビットアナログデジタル変換器２３８（例えばマイクロフォン１０からの音声をボード１９７を介してデジタル化するのに用いられる）と、フロッピーディスク操作器２４０と、リアルタイムクロック２４２とに結合される。

【００３２】

メインＣＰＵ２２８およびゲームＣＰＵ２００は、別個に起動する。電源が入ると、ゲームＣＰＵ２００は、最初リセット状態になり、その後再開される。ゲームＣＰＵ２００は、初めにモニタＲＯＭ２０４からのプログラム命令を実行する。メインＣＰＵ２２８は、ゲームＣＰＵ２００がアクセスするメモリ領域を制御する。メインＣＰＵ２２８は、ＣＰＵ２００がモニタＲＯＭ２０４の中から実行するべきか、それともＲＡＭカートリッジ４の中から実行するべきかを識別するゲートアレイ２２６にレジスタを格納する。メインＣＰＵ２２８は、ＲＯＭ２３４内の初期プログラム格納およびオペレーティングシステム命令を実行し、フロッピーディスクドライブ１９９およびフロッピーディスク操作器２４０を介して、モデルゲームソフトウェアを含むフロッピーディスク８の内容を読み出す。モデルゲームソフトウェアに関連する操作を制御するためのオペレーティングシステムプログラム命令は、メインＣＰＵ２２８が実行できるようにＤＲＡＭ２３０に転送される。システム電源を切って、その後入れると、メインＣＰＵ２２８は、フラッシュメモリ２３６を検査する。最初に電源が投入されると、オペレーティングシステムプログラムがフロッピーディスク８から読み出されて、フラッシュメモリ２３６に格納され、ＤＲＡＭ２３０に転送される。２回目に電源が投入されると、オペレーティングシステムプログラムは、フロッピーディスク８を用いることなく、フラッシュメモリ２３６からＤＲＡＭ２３０に転送される。通常、フラッシュメモリのアクセス速度は、フロッピーディスクのアクセス速度よりも高速である。従って、システムは、２回目の電源投入時には、すばやく開始できる。

【００３３】

フロッピーディスク操作器２４０からのモデルソフトウェア情報は、まず、ＲＡＭカートリッジ４へ転送するために、ＤＲＡＭ２３０内にバッファリングされる。その後、メインＣＰＵ２２８は、通信可能な信号を生成して、その信号をゲートアレイ２２６内のハンドシェークポートに送信する。ゲームＣＰＵ２００は、モニタＲＯＭ２０４に記憶されている命令を介してハンドシェークポートを監視し、ゲートアレイ２２６内の通信ＲＡＭからのデータおよび／または命令を受信する準備をする。ハンドシェイクポートは、一対の一方向ハンドシェイクポートによって実施されてもよい。一方の一方向ハンドシェイクポー

トは、情報をメインＣＰＵ２２８からゲームＣＰＵ２００へ伝達する。他方の一方向ハンドシェイクポートは、情報をゲームＣＰＵ２００からメインＣＰＵ２２８へ伝達する。

【００３４】

ハンドシェイクポートは、情報がメインＣＰＵ２２８とゲームＣＰＵ２００との間で転送されるべきかを示すゲームＣＰＵ２００によってアドレス指定可能なバッファレジスタを含む。ゲームＣＰＵ２００は、ゲートアレイ２２６内の通信ＲＡＭがメインＣＰＵ２２８から情報を受信したことを決定すると、通信ＲＡＭからの情報にアクセスし、８０ピンコネクタ１８９およびコネクタボード１８６を介して、そのような情報をＲＡＭカートリッジ４へ転送する。メインＣＰＵ２２８は、通信ＲＡＭ情報がＲＡＭカートリッジ４に転送された後に、ゲートアレイ２２６内に実施されたレジスタをセットして、ゲームＣＰＵ２００の制御をモニタＲＯＭ２０４からＲＡＭカートリッジ４へ切り替える。ゲームＣＰＵ制御がＲＡＭカートリッジ４に切り替えられた後は、ゲームＣＰＵ２００は、もはやハンドシェークポートを監視しないか、またはゲートアレイ内の通信ＲＡＭとのさらなる対話を行うことはない。この時点で、ゲームＣＰＵ２００が動作して、上述のＳＮＥＳ関連特許に記載のゲーム機と実質的に同様の方法で、ＲＡＭカートリッジ４からゲーム関連命令を実行する。

【００３５】

メインＣＰＵ２２８は、ゲートアレイ２２６内のレジスタ複写ＲＡＭにアクセスすることによって、ＣＰＵ２００システムバスを介して送信される情報を常に監視する。ゲームＣＰＵ２００バスからレジスタ複写ＲＡＭに書き込まれた情報は、メインＣＰＵ２２８フォーマットと互換性のあるようにフォーマットされる。ＰＰＵ２１２および音声プロセッサ２０８からの情報は、メインＣＰＵ２２８が監視メインをしてゲーム状態を継続的に認識できるようにするために、レジスタ複写ＲＡＭへ流れる。３２ビットＲＩＳＣプロセッサを用いれば、そのような大容量の情報を監視するのに必要な処理能力がメインＣＰＵ２２８に提供される。後述するように、ＰＰＵ２２４とビデオＲＡＭ２２０とを介してメインＣＰＵ２２８によって生成された画像と、ＰＰＵ２１２とビデオＲＡＭ２１４とを介してゲームＣＰＵ２００によって生成された画像とは、一部がゲートアレイ２２６にある後述の重ね合わせ操作器２１６を用いることによって、ユーザのテレビ１５上に同時に表示される。

【００３６】

メインＣＰＵ２２８は、ゲームＣＰＵ２００の操作を継続的に認識しているので、全ての要求される情報が表示操作のためにビデオＲＡＭ２１４に書き込まれて初めて、ゲームプレイを修正するためにユーザが命令した「システムブレイク」を開始する。システムブレイク論理回路は、システムブレイクコマンドが実行された場合に、ゲームブレイク制御信号をゲームＣＰＵ２００に対して生成するためのタイミングを制御する。このように、メインＣＰＵ２２８は、ゲームＣＰＵバス上のデータ、特にビデオ表示または音声の生成に関連する情報を監視して、システムブレイクを実行する適切な時点を決定し、それによって歪んだピクチャおよび音声情報を回避する。

【００３７】

図５および図６は、情報がフロッピーディスク８からＲＡＭカートリッジ４に転送された後の図３および図４に示すメモリ装置のうちのいくつかのメモリマップの一部である。フロッピーディスク８に記憶されたモデルソフトウェアは、制御ファイル、モデルゲーム関連データ、ユニットポインタ、アドレスデータおよびメインゲームプログラム（および、図示していないが、音声プロセッサ２０８に入力される音声データ）を含む。図５からわかるように、そのようなモデルソフトウェアの全てがＲＡＭカートリッジ４に転送されるわけではなく、むしろＤＲＡＭ２３０は、編集ソフトウェア（例えばマリオファクトリー）およびオペレーティングシステムソフトウェアと同様に、ユーザが選択したとおりのゲーム条件変更に関するデータを記憶する制御ファイルを保守する。モデルソフトウェアが実行されると、ユニット作業用データがモデルソフトウェアのメインプログラムによって生成される。ユーザがゲームプレイを停止してモデルゲームを修正するために「システム

ブレイク」を開始した後、ユニット作業用データと、ユニットポインタと、編集用データとが、バッファ領域へ転送される。制御ファイルを用いたエディタが終了すると、修正された情報（後述のデータおよびユニットポインタ情報）は、バッファ領域データに記憶されて、その後ゲームメインプログラムを記憶するＲＡＭカートリッジ４に転送される。

【００３８】

本発明のゲーム処理システムは、様々なネットワーク構成で使用する目的で設計される。本発明は、「友だちネットワーク」、すなわちユーザ間で新規に生成されたビデオゲームを記憶したＲＡＭカセット４を交換することを考慮している。同様に、識別カード６およびフロッピーディスク８は、同一のゲーム処理プロセッサシステムにおいてＲＡＭカセット４を用いて使用する目的で、友人の家に運ぶこともできる。

【００３９】

さらに、図７に示すように、オンラインネットワークシステムも考慮している。図７に示すように、ゲーム情報は、図１に示すモデム２２を介して、電話回線４３によって、ユーザの家４８からアクセスポイント４２、デジタルリンク４７を介して、ゲーム処理センタ４０へ送信することができる。同様に、ゲーム情報は、電話回線４３を介して友人の家５０からアクセスポイント４４に転送されて、その後デジタルリンク４７を介してゲーム処理センタ４０に送信することができる。ゲーム情報は、また、ゲームメーカ４６（例えば任天堂など）から送信することもできる。図７に示すオンラインネットワークシステムにおいては、ユーザのフロッピーディスク８内のモデルソフトウェアを、修正すべきモデルビデオゲームソフトウェアに付け加えるために容易に補足することが可能である。

【００４０】

本発明の一つの典型的な実施例によると、フロッピーディスク８に記憶されたモデルソフトウェアは、メーカによって設計され、ユーザが変更することのできないビデオゲーム部分を含む。モデルソフトウェアのこの部分は、「ベースファイル」と呼ばれる。「ユーザファイル」は、ユーザが変更できるビデオゲーム部分である。図７に示すシステムにおいては、ユーザファイルは、ゲーム処理システムを持つ友人の家にネットワークを介して送信することができ、ユーザ間で対話型ゲームプレイをしたり、または友人が、新規に設計されたゲームの修正バージョンをプレイすることができる。アクセスポイント４２および４４は、例えば、ゲームを非常に遠方へ送信するのにかかる電話料金を最小限にするために供給される。その後、ゲームデータは、直接デジタルリンク４７を介してゲーム処理センタ４０に伝送される。ゲームデータは、ゲーム処理センタ４０で受信された後に、そこから適切な目的地まで再送信される。

【００４１】

ゲームプロセッサユニットシステムハードウェアおよびソフトウェアをさらに詳しく説明する前に、実施例のゲーム作成および編集操作について次に説明する。フロッピーディスク８がコンソールソケット７に挿入されて、ゲーム処理識別カード６がソケット５に挿入された後に、タイトル画面が、例えば図８に示す典型的な画面の「マリオファクトリー」のように表示される。図８に示すタイトル画面は、ユーザがマウス１６によって選択することのできる５個のアイコンを示す。アイコンは、「クリックする」、すなわち左のマウスボタンを軽く押圧することによって選択される。図８において５個のアイコンが示されているが、以下の説明は、多くの可能な実施態様の内の一つを表す例示的な実施例である。例えば、５個より多いかまたは少ないアイコンをタイトル画面に表示してもよいし、また、異なる機能を、例示的な実施例に示すものに付け加えるか、またはそれらの代わりに選択してもよい。

【００４２】

タイトル画面は、「モデルソフトウェア格納」アイコンを示し、それがマウス１６によって選択されると、フロッピーディスク８に記憶されたモデルソフトウェアが、より詳細に後述する方法によってコンソールユニット２のメモリに格納される。図８は、また、「ゲームアイコン」を含み、そこでモデルソフトウェア内のゲームを実行および修正するために選択することができ、新規かつモデルソフトウェア内のゲームとは全く異なるゲームを

作成することができる。図８に示すタイトル画面は、また、電話回線を介して目的地までデータを送信するためのモデム２２を選択することのできるネットワークアイコンを含む。さらに、ディスク退避およびツール編集アイコンも選択することができる。

【００４３】

図８に示すゲームアイコンがクリックされた場合、通常、ゲームプログラム開始画面が表示される。ユーザは、ゲームを一人のプレーヤでプレイするか、または二人のプレーヤによってプレイするかを選択した後に、図１に示す操作器１２または１４の開始ボタンを押す。次に、例えば周知の「マリオブラザーズ」の内の１つのバージョンなどのモデルソフトウェアによって規定されたゲームのプレイを開始する。

【００４４】

ゲーム中、表示された移動オブジェクトキャラクタまたは表示画面背景の一部を変更したいと所望する時はいつでも、ユーザはゲームを停止して、本願で「システムブレイク」と呼んでいるものを開始する。システムブレイクが開始されると、図９に示すようにシステムブレイク画面が表示され、その中でシステムウインドウ画面５０が、ユーザが選択したゲーム画面５７上に重ね合わせられる。例えば、本出願人が製造したビデオゲームの中で有名な周知の「マリオ」などの移動オブジェクトキャラクタ、または背景のどの部分でも、変更がなされるべき領域においてクリックすることによって修正される。

【００４５】

本発明は、モデルソフトウェアに対して広範囲の変更を行うことができ、モデルソフトウェアゲームの背景および移動オブジェクトを、モデルソフトウェアゲームの原型が認識できない程までに修正することができる。例として、図９は、カーソル（例えば手）が修正すべきキャラクタに重なった時にユーザがマウスをクリックすることによって、「マリオ」と認識できる移動オブジェクトキャラクタに変更がなされることを示している。所望するならば、全く新しいキャラクタをマリオの代わりに用いることができる。

【００４６】

この例として、ゲームをしているユーザが、マリオの口髭を取り除きたいと望んでいると仮定する。上述したように、ユーザはマウスをクリックして、システムブレイクを開始することによってゲームを停止する。その後、ユーザはマリオをクリックする。メインＣＰＵ２２８は、例えばキャラクタの「ユニット番号」の表示５２と共に、変更すべきキャラクタを線で囲む矩形ブロックを生成する。修正することのできる各移動オブジェクトおよびゲームの背景部分は、それぞれ特有のユニット番号によって識別される。後に詳細に説明するように、各ユニット番号およびそのオブジェクトに関連するものは、広範囲の特性、例えば大きさ、ポーズ、動画パスなどである。システムウインドウ５０は、ユーザが広範囲の編集機能を選択できるようにする一連のアイコンを含む。システムウインドウ「ステージ」アイコン５４は、広範囲の背景キャラクタ編集を行う図２３に示す「ステージ」ウインドウのアクセスを可能にする。システムウインドウは、また、タイトル画面アイコン５５への復帰と、ゲーム再開アイコン５９と、ユーザがゲーム停止とスロープレイモードとの間を切り替えることができるようにする切替可能制御５３とを含む。選択しうる特定のゲーム作成／編集ツールをシステムウインドウ５０において入れたい場合は、多くの追加または代替の編集機能をシステムウインドウ５０内に含んでもよい。

【００４７】

図９からわかるように、ユニット番号表示５２と同様に、アイコンを表示するシステムウインドウ５０および編集ツールは、ゲーム画面５７上で重ね合わされる。編集のためにマリオキャラクタを選択すると、一連のマリオ編集画面／ウインドウが表示される。他の移動オブジェクトは、同様の方法で編集のために選択することができる。

【００４８】

図１０は、典型的なキャラクタ編集画面を表している。図１０は、マリオキャラクタが修正されている（６０）ことを視覚的に示すマリオキャラクタ編集関連ウインドウを示す。マリオの表示の下には、マリオの現在のステータスが確認される（６２）。「マリオ」のステータスは、「停止」していると表される。確認可能な他の典型的なステータス表示は

、マリオ「左移動」、マリオ「右移動」、マリオ「ジャンプ」の表示、もしくはマリオが上方または下方から現れることである。後述のステータスエディタを用いることによって為しうるマリオのステータスの変更に従って、ゲームは変更される。

【００４９】

ステータスブロックの下に、一連のアイコンが示され、なしうる典型的なマリオ関連編集修正と、関連アイコンをクリックすることによってアクセス可能な編集画面とを示す。アイコン６４は、図１４と共に後述するように、マリオの動画パターンに関して行われる変更が行われるようにする。アイコン６６は、図２２と共に後述するように、キャラクタ「ステータス」条件に対して変更がなされるようにする。アイコン６８は、マリオが移動する時に、ビデオゲームに関連する効果音に対して変更がなされるようにする。アイコン７０は、ゲームプレイを再開する。アイコン７２は、図１２と共に後述するように、「全ステータス」条件に対して変更がなされるようにする。アイコン７４は、図１１に関連して後述するマリオキャラクタドットパターンに関して変更がなされることを可能にする。アイコン７６は、図２１に関連して後述するキャラクタ動作の編集を可能にする。アイコン７８は、システムをシステムウインドウへ復帰させる。

【００５０】

例えば、マリオの口髭を取り除くためには、ユーザは、キャラクタアイコン７４をクリックし、図１１に示す表示画面を生成する。キャラクタアイコンをクリックすることによって、制御は、図１１に示すキャラクタ設計ツールに転送される。ズームツールは、アイコン８５上でクリックすることによって選択され、修正すべきキャラクタの拡張図が現れる。ペンシルツールは、ペンシルモードとボックスモードとを切り替えるアイコン８６をクリックすることによって、選択される。ペンシルモードでは、グラフィックデータをドット毎に変更することができる。ボックスモードでは、グラフィックデータを矩形領域によってペイントすることができる。アイコン８４は、ゲームプレイの再開またはシステムブレイク画面への復帰を制御する。消去ツールは、例えばマリオのパターンに関するすべてのグラフィックデータを消去するアイコン８７をクリックすることによって、選択される。アンドゥツールは、直前のコマンドを取り消すアイコン８８をクリックすることによって、選択される。図１１に示すように、カラーパレット８０は、ユーザが多くの色の内のどれでも選択して、選択した色をクリックすることによってマリオのどの部分の色でも変更し、従来のペインティングプログラムに従って彩色ツール（例えば、クレヨン型カーソル）を用いてマリオを描くことができるようにする。このカラーパレット機能は、ユーザが、規定のカラーパレット内の色を選択したり、色の様々なパレットの中から選択したり、カラースペクトル機能を用いて新規の色を作成することを可能にする。

【００５１】

モデルソフトウェアに関連する各移動オブジェクトは、所定の数の異なるポーズを有する。例として、マリオパターンは、合計１３のポーズパターンで構成されている。パターンの数は、図１１（８２）に示す画面の左部分に表されている。例えば、パターン番号ゼロは、「０」をクリックすることによって選択される。ペンシルモードがアイコン８６をクリックすることによって選択されると、クレヨン型のカーソルが、パターン番号０のマリオの口髭を消去するために用いられ、カラーパレット８０のマリオの肌の色と置換される。同様の操作が、残りのマリオパターン１から１２のそれぞれを用いて遂行される。

【００５２】

口髭の全てがいったん消去されると、例えばロボットアイコン８４が、システムウインドウへの復帰を生成するためクリックされ、システムウインドウ復帰においては、ゲームアイコンがゲームプレイを再開するために選択される。ゲームプレイが再開されると、マリオは、ゲームの間中口髭無しで表示される。図１１に示す表示画面を用いて、ユーザは、マリオパターンの各画素を修正することができる。ユーザは、システム内の利用可能な様々なエディタを用いて、独自のマリオを作成し、広範囲の動画およびマリオまたは他のキャラクタ上で遂行される他の操作を制御することができる。

【００５３】

図１２は、「マリオ」という名のユニット用の例示的な「全ての」ステータス画面エディタ表示を示す。上述のように、各移動オブジェクトまたは他の修正可能なキャラクタは、ユニット名によって識別される。各ユニット名は、関連するステータスの集合体を有する。本発明は、キャラクタのステータスを識別する「全ステータス」エディタを利用し、ユーザが操作を選択して、ステータス修正をすることを可能にする。このようにして、図１２に示すように、ユニット名９０、すなわちマリオ用に、「停止」、「左移動」、「右移動」、「左滑り」、「右滑り」、「ジャンプ」、「落下」、「つぶれる」など（９２）の様々な識別されるステータスがある。さらに、ステータスを複写、または削除するためのツール９４がある。オブジェクトの条件がゲーム実行ＣＰＵ２００によって操作されるため、識別されたステータス９２は、オブジェクトの条件をオペレーティングシステムＣＰＵ２２８に伝える。各ステータスに関しては、システムＣＰＵ２２８には、図５に示す制御ファイルデータ構造を介して、各ステータスに関連する操作が伝えられる。このように、各ステータスに関して、関連する操作のシーケンスが識別および記憶される。例えば、「ジャンプする」というステータス５に関して、データ構造は、ジャンプ中に生成された音声（例えば「ポヨーン」）、ジャンプ中の動作（例えば、動画プログラムによって識別された動画パス）、およびジャンプ操作中のプログラミング（マリオが床にぶつかったら、キャラクタが死ぬ）の指示を記憶する。

【００５４】

図１２左側のステータス見出しは、ステータスに関連する操作のより詳細なシーケンスへのアクセスを可能にする。例えば、図１２に示すように、ステータス１項目が図１２の右部分においてユーザによって選択された場合、ステータスに対する動画が表示される。さらに、ステータスに関連する効果音が生成されて、キー、オート、効果音およびアニメエントリのもとで適切なエントリが表示される。図１２に示すように、「キー」エントリが、特定のステータスに対して動作中の操作器がプレーヤ１用か、またはプレーヤ２用か、およびどのボタンがアクティブなボタンかを識別する。「キー」アイコンが選択されると、図１３と共に後述するように、キーエディタが用いられる。「オート」エントリは、識別されたステータスに関して図２１で説明するオートプログラマの状態を表す。効果音およびアニメは、それぞれ、効果音および動画を識別する。さらに、背景音楽ＢＧＭも、効果音エントリで選択可能である。

【００５５】

ゲームが停止され、ユーザがマリオをクリックすると、システムＣＰＵ２２８は、図１２のステータス表示に対応する制御ファイルにアクセスして、選択されたどのシーケンスが、識別されたステータスと関連するべきかを知る。このようにして、識別されたゲームステージにおいて特定の画面と関連する情報は、システムＣＰＵ２２８に転送される。

【００５６】

キー機能ステータスが図１３で説明されるキーエディタによって制御される。キーエディタは、操作器１または操作器２に関連するステータス、操作器上のあらゆるボタンおよび／またはボタンが（一度）押されたか、ボタンが（継続的に）押されたかというようなボタン状況を定義する。図１３に示すように、キー名表示は、キー名によって一般的な型のキー（例えば８）をセットアップすることができる。リネームボタンは、キー名を変更することができる。ゲームオプションネームは、ステータス編集のようにメニューからオプションを選択することができる。キー状況表示によって、ゲームオプションが、キー条件が一致したときのみ実行されるようになる。１Ｐ／２Ｐアイコンをクリックすると、１Ｐまたは２Ｐ操作器を選択することができる。操作器表示は、このボタンをクリックすることによって、「キー」か「（複数の）キー」を選択することができる。

【００５７】

図１０において動画アイコン７６が選択されると、動画エディタが選択されて、例えば図１４に示すような動画画面を表示する。図１４は、動画シーケンスを作成する様々なポーズのうちの蟹の形をした移動オブジェクトを示しており、動画シーケンスにおいては、各ポーズがポーズ番号が割り当てられている。図２１に関連して後にさらに説明するように

、「オートプログラマ」ツールは、移動が起きるパスを定義するために用いられる。

【００５８】

図１４に示す動画ツールは、ポーズのシーケンスを定義する。本実施例においては、８個のポーズを動画シーケンス内で定義することができる。図１４は、画面に表示すべき動画シーケンスの中で、８個のポーズのうち４個（ポーズ番号０から３までが付けられている）に対する領域を示す。ユーザは、また、操作器１０２を用いて各セルが現れる時間を定義する。セル時間表示１０４が６０に設定された場合、各セルは１秒間表示される。シーケンスの全体のテンポは、テンポ制御１０６を修正することによって制御することができる。本実施例において、テンポの標準速度はゼロに設定される。テンポを遅くするためには負数を設定し、テンポを早くするためには正数を設定することによって、テンポを制御することができる。ゲームプレイが再開されると、各セル表示頻度および動画シーケンス速度は、図１４の設定に基づいて制御される。

【００５９】

図１４におけるポーズ８から１５（まとめて１０８）は、動画シーケンスに挿入することのできる代替のポーズを識別する。ユーザは、特定のポーズ、例えば１０をクリックし、そのポーズをドラッグして、例えば動画シーケンスポーズ２上に配置するだけでよい。この操作により、ポーズ１０がポーズ２に替わって表示される。

【００６０】

図１４の上方左隅に示す四隅の印によって定義される領域は、特定のキャラクタに対する衝突しきい値１０９を表す。すなわちゲーム中に、識別されたキャラクタが、他のオブジェクトに対して四隅印の領域によって示された距離内に現れた場合、衝突が検出される。四隅印の領域の大きさは、「ヒット」表示制御しきい値制御１１２によって制御することができる。

【００６１】

図９に戻って、システムブレイクが開始された後に、ユーザがシステムウインドウ５０のステージアイコン５４をクリックした場合、図２３に示すステージウインドウが画面上に表示される。「画面エディタ」が、図１５に示すように、ステージウインドウの画面アイコン１５２上をクリックすることによってアクセスされる。所望するならば、画面エディタを用いて、背景要素を図１５の底部から背景画面部分にドラッグし、実際の表示背景を修正することができる。画面エディタは、ユーザが、システムブレイクの際にゲームに関連する多くの異なる背景層のいずれか１つを検討し、修正することができるようにする。画面エディタは、背景を水平および垂直にスクロールするための制御を含み、ユーザが、例えば第１、第２または第３の背景画面の選択が可能な、ユーザがアクセスできる「一覧」モードを介して全背景を見ることができるようにする。それぞれの異なる背景層は、スクリーンエディタを介してアクセス可能であり、スクロールは、各背景層の全体を見ることができるように、また各層を修正することができるように制御することができる。

【００６２】

画面エディタは、関連の「マップエディタ」を有し、マップエディタは、図１６に示すように開くことが可能である。マップエディタは、マップエディタにおいて減少された形で示される利用可能な背景画面を再配置させるために用いられる。マップエディタの「貼り付け」機能を用いて、ゲームの背景表示画面は、例えば図１７の１２０に示されるようなキャラクタが描かれた背景画面をドラッグして、アクティブな背景上にそれを貼り付けることによって修正することができる。

【００６３】

ビデオゲーム背景に関連する音楽は、「ステージ」ウインドウに関連する音楽記号アイコンをクリックすることにより、音楽エディタにアクセスして修正することができる。「ステージ」ウインドウは、後でさらに説明するが、図１８に示すような画面表示になる。図１８に示すように、音楽エディタはゲーム楽譜を表示する。（音楽編集中にはカラーで表示される）「スイッチ」は、ゲーム楽譜表示上の同じ様に色付けられた数字に対応する。図１８に示す表示されたスイッチの内の１つをマウスで選択することによって、音楽の低

音部分、音楽のピアノ部分、または選択される他の楽器の部分のみを録音再生することが可能である。低音、ピアノおよび他の音楽音を選択的に重ねることによって、背景音楽が作られる。

【００６４】

図１８において、カラー記号がクリックされて、選択された音楽セグメントが識別されると、図１９に示すウインドウが生成されて、図１８の音楽エディタ画面に重ね書きされて、それによって、選択された数の各部分に対して、音楽トーン、音量等の個別制御が可能となる。例えば、図１９に示す「キー」制御が操作されると、背景音楽キーは、マウスが移動してキー制御を別のセッティングにドラッグするにつれて、変更される。同様の制御が、「エコー」「テンポ」「パン」および他の所望する背景音楽の局面に対して実行される。「パン」制御は、左／右ステレオのバランスを操作することができる。

【００６５】

さらに、本発明の音楽エディタは、音楽の再生中にリアルタイムでその音楽を変更することを可能にする。プレイ記号がクリックされると、背景音楽が鳴って、図２０に示すウインドウが生成される。ウインドウは、「音量」「テンポ」「エコー」「キー」を、音楽の演奏と同時に変更するために用いられる。例えば、「キー」アイコンをクリックすると、図１９のキーバーのような制御バーが生成されて、リアルタイム制御が実行される。

【００６６】

元のモデルソフトウェアは、ゲームに関連する元の音波形を所定の数、例えば３２個含んでいる。ユーザは、元の音波形を８つの音コースに割り当てることができ、音楽エディタを用いて音コースのどのような面でも選択的に修正することができる。楽譜上の各音コースを識別する数字は、元の音波形のそれぞれと関連する独自の印によって表示されてもよい。音楽エディタソフトウェアは、図２０に示す各アイコンの状態を継続的に検査して、それに従って出力音楽を修正する。さらに、マイク１０から記録されたサンプリング波形を元の音波形リストに追加することも可能である。

【００６７】

ゲーム処理システムは、「オートプログラマ」機能をさらに含んでおり、それによってユーザは、オブジェクトの動作を修正することができる。オートプログラマの操作を説明する典型的な表示が図２１に示されている。オートプログラマは、図１０に示すオートプログラマアイコン７６をクリックするか、または図１２の「全ステータス」エディタに関連するオートプログラマアイコンによってアクセスすることができる。初めに、図２１に示すキャラクタのようなユニットが、左マウスボタンを押圧することによって選択される。マウスを用いて、点で示される最大６４個までの動作を含むパスを記憶することができる。マウスがまだパスに沿って移動している間に左マウスボタンを押すと、全パスが記憶される。点同士が近づくように点を配置すると、動作は遅くなる。点同士が離れると、動作が速くなる。しかしながら、動作の速度は、図２１に示すテンポ制御を用いることによって調整することができる。図２１は、関連プレイまたは停止制御を有しており、その中でパスを再生したり、またはユーザが所望する場合は、パスを停止したりすることができ、その後パスが変更される。図２１に示す「効果音」ボタン１３０およびステップ制御バーを用いて、オブジェクトの動作に音声を関連させることができる。例えば「バン」という音声を、パスの頂点に達したオブジェクトに関連させることが可能である。ステップボタンを保持することによって、オブジェクトを音声が付け加えられるべきポイントまで移動することができる。「効果音」ボタンをクリックすることによって、選択した音声をオブジェクトが停止するまでのパス内のポイントに関連させる。

【００６８】

ゲームプロセッサシステムは、また、「ステータス」エディタを含む。ゲーム中、例えばキャラクタのマリオがジャンプしている時にシステムブレイクが開始された場合、図１０に示すマリオウインドウがアクセスされる。「ステータスエディタ」アイコン６６がこの点でこの時に選択された場合、図２２に示すような表示画面が現れる。図２２において、「ジャンプ」というステータスを有するキャラクタのマリオに対して、第１のコラム１４

０および第２のコラム１４２を有するステータス画面が示される。コラム１４０は、条件のシーケンスを定義する列を含む。条件は、例えばマリオがジャンプした時に特定の高さに達するかどうかを規定する。コラム１４２は、各条件に対して、その条件が検出された場合起きる結果関連処理を識別する。例えばマリオがジャンプした時に一定の高さに達する場合、マリオは、例えば消える。ステータスエディタは、ユーザが、条件および／または結果を変更することができるようにする。例えば、ユーザは、マリオがジャンプして一定の高さに達する結果を、「マリオが消える」から、例えば「何も起こらない」に変更することができる。その後ゲームが再開されると、修正された処理は、条件が検出された時に遂行される。図２２に示すステータスエディタは、また、図１に示すキーボード１８を介して、ビデオゲームプログラミング言語命令に通じた人が、新規の処理を作成することを可能にする。コラム１４２の処理をクリックすると、処理リストが画面上に生成される。リストにある処理をどれでも選択して、変更することができる。処理リストの処理をクリックすると、処理の処理プログラムが、ビデオゲームプログラム言語を用いて修正される。ビデオゲームプログラム言語命令が自動的にコンパイルされて、処理リストの新規の処理として格納される。

【００６９】

ゲームプログラム言語命令を直接入力することは、ＢＡＳＩＣ命令に似通っており、上級のゲーム設計者によって用いられるようになっている。そのようなゲームプログラミング言語命令は、周知のゲームジャンルに基づいてモデルソフトウェアからゲーム設計を始めることを選択しない上級のゲーム設計者が、代替的に用いることができる。すなわち、そのような上級のユーザは、キーボードを使用して、自分の独自のモデルソフトウェアを作成し、本件に記載する機能を使用して、自分の作成した独自のゲームを修正する。

【００７０】

図９に戻って、上述したように、システムウインドウ５０は、ステージアイコン５４を含む。ステージアイコンは、図２３に示すステージウインドウを開く。ステージウインドウ表示１５０は、現在のステージ番号、および例えば「ゲーム中」などのステータス名を表示する。アイコン１５２は、画面ボタンであり、先に説明したように、現在の背景の編集を行う。上述したように、マップアイコン１５４は、現在の背景マップの変更を行う。「ＢＧＭ」と表示されたアイコン１５６を選択すると、現在の背景音楽を編集できる。ステータスアイコン１５８は、現在のステージの上述のようなプログラミングを変更するために、ステータスエディタにアクセスする。アイコン１６０は、図９に示すシステムウインドウに戻る。アイコン１６２は、ゲームプレイを再開し、アイコン１６４は、現在のステージ番号および現在のステータス名を表示する。

【００７１】

図２４から図２６までは、「マリオファクトリー」画面から始まって、図８から図２３に関連して上述したゲーム修正／編集表示画面にアクセスする方法を要約している。図２４は、図８に示すマリオファクトリータイトル画面を機能的に表示したものである。図２４に示すファイルのアイコンは、ファイルの選択を示す。ユーザが「ツールボックス」アイコンをクリックすると、図２５に機能的に示すツールボックス選択画面および編集機能がアクセスされる。「ゲーム」アイコンをクリックすることによって、図２６に示すモデルソフトウェア関連編集機能がアクセスされる。「構成」アイコンをクリックすると、ユーザは、マウスまたはキーボードの構成のいずれかを変更することができる。ネットワークアイコンをクリックすると、（ゲームの修正可能部分を表示する）モデルソフトウェアおよび／またはユーザファイルが、モデム２２を介して他のユーザへ送信されるか、または他のユーザから受信される。

【００７２】

図２５は、「ツールボックス」選択画面を介してアクセス可能な、編集関連ツールを示す。ツールボックス画面をアクセスすることによって、ゲームプレイをゲームコンテキスト外で変更することができる。このように、ユーザは、特定のキャラクタまたは所望されるゲームの他の面を変更するために、全ゲームをプレイする必要はない。代わりに、ユーザ

は、データに直接進んでそれを変更することができる。ツールボックスアイコンをクリックすることによって、図８のマリオファクトリータイトル画面からアクセスされるツールボックスを使用することにより、上級のユーザが、ゲーム自体のコンテキストから変更を行わうことなく、ゲームに多くの詳細な変更を加えることができる。「作者から」ブロックは、作者が、ネットワークを介して、例えば送信されたゲームプログラムの詳細を記述しているメッセージを送信することができることを示す。「マップコレクション」ブロックは、様々な背景マップを収集して、画面エディタ、ユニット画面エディタおよびマップエディタを用いてこれらのマップを編集することが可能であることを示す。ユニット画面エディタは、アクセスされて、ユニットをマップ上の指定の位置に配置する。通常、ユニット画面エディタは、図２６に示すマップエディタ上の「ユニット」アイコンをクリックすることによってアクセスされる。ツールボックス選択画面では、ユニット画面エディタは、マップコレクションメニューから直接アクセスされる。

【００７３】

「ツールボックス」選択画面を介して、「プログラム収集」機能をアクセスして、アセンブラリスト、基本リストおよびオートプログラマリストを受信するようにしてもよい。基本リストおよびオートプログラマリストは、それぞれ、プログラムエディタおよびオートプログラマによって編集される。

【００７４】

「ユニットコレクション」機能は、ゲーム中に存在する様々なユニットを識別するユニットリストにアクセスするために用いられることができる。ユニットリストは、全てのステータスエディタによって編集することができる。リストは、ゲームプログラム内に登録されている全てのユニットをコンパイルする。さらに、ツールボックス画面は、音楽エディタによって編集可能な背景音楽リストへのアクセスと、特殊効果エディタによって編集可能な特殊効果（ＳＥ）リストへのアクセスとを提供する音声コレクション機能のアクセスを行うことができる。

【００７５】

ツールボックス選択画面は、また、「ゲーム構造」コレクション機能へのアクセスを行う。これらの機能は、一般的に全ゲームへ適用され、個々のユニット適応性を問わない。例えば、「ゲーム構造」機能は、特定のキャラクタがゲームが終わる前に何回死ぬかを識別することができる。この特徴は、ゲーム構造編集中に設定される。「ゲーム情報」機能に関しては、このカテゴリーのゲーム構造は、緑亀がゲームに何回現れるか、または最初の画面レベルが何かを識別することができる。そのような情報は、データエディタを介して編集することができる。ステージリストゲーム構造情報は、利用可能な様々なゲームシーケンスに関連する。ステージリスト情報は、ステータスエディタ、プログラムエディタまたはデータエディタを介して修正することができる。

【００７６】

図２６は、ゲームプレイ中に行うことのできる編集／ゲーム作成変更を示す。システムブレイクが開始された後、詳細に上述したように、ユーザは、編集のために移動オブジェクトユニットを選択することによって、または背景を変更するためのステージアイコンを選択することによって、エディタをアクセスすることができる。示される様々なエディタは、様々な画面表示に関連するアイコンに対応している。例えば、動画エディタ、ドットエディタ、オートプログラマ、特殊効果エディタ、全ステータスエディタ、キーエディタ、ステータスエディタ、プログラムエディタおよびデータエディタは、例えば図１０に示すアイコンを用いてアクセスすることが可能である。同様に、例えば音楽エディタ、ステータスエディタ、画面エディタ、ユニット画面エディタ、マップエディタおよびデータエディタなどのステージまたは背景関連エディタは、図２３に示すステージウインドウ関連アイコンと関連している。典型的な実施例として図２６に示すように、ユニット画面エディタは、画面エディタまたはマップエディタのいずれかを最初にアクセスすることによって、到達可能である。他のエディタを介してアクセス可能なエディタは、通常、あまり頻繁には利用しないエディタであり、より上級のゲーム開発者が、例えばゲームプログラム言

語命令を実際に変更するためにプログラムエディタを使用するようなエディタである。
【００７７】
図２５および図２６に示すエディタは、図８から図２３に関連して説明した方法で動作する。例えば、図２５および図２６に示すオートプログラマエディタによって、図２１に示す表示画面上に示されるようにオブジェクト移動を編集することができる。図２５および図２６に示すステータスエディタは、図２２の画面表示に関連して説明した方法で動作する。図２５および図２６に示す動画エディタは、図１４に関連して説明したように動作する。図２６に示す全ステータスエディタは、図１２の表示画面に関連して先に説明したように動作する。
【００７８】
本発明の実施例に関連するハードウェアに戻って、図２７は、図３に示すゲートアレイ２２６のより詳細なブロック図を示す。ゲートアレイ２２６は、ＣＰＵ２２８およびゲームＣＰＵ２００からのビデオ信号出力を重ね合わせる機能を含む様々な機能、ゲーム処理デバッキング関連機能（ハードウェアおよびソフトウェア中断、アドレス、データトラップなど）、および様々なインタフェイス機能（例えばゲームＣＰＵ２００／メインＣＰＵ２２８インタフェイス制御、ゲーム手動操作器インタフェイスおよびマウス、キーボードおよび識別カードインタフェイス）を制御する役割を果たす。ゲートアレイ２２６は、また、ＣＰＵ２２８バスアクセス制御、割り込み制御、ＣＰＵ２２８実行可能制御、ＤＲＡＭ制御に関連しており、ＤＲＡＭ制御には、制御バス制御と、アドレス多重化と、リフレッシュ制御と、ＲＯＭ２３４、フラッシュメモリ２３６、フロッピーディスク操作器２４０等の制御を含む周辺装置ＬＳＩアクセス制御とが含まれる。
【００７９】
図２７に示すように、ゲートアレイは、ＣＰＵ２２８，ＣＰＵ２００，ＰＰＵ２１２などを含む、図３および図４に示す様々な構成要素と相互動作する。アドレスデコーダ２６０は、ＣＰＵ２２８のアドレスバスに結合され、ＣＰＵ２２８に関連する他の装置へのチップ選択信号を生成する。図２７においてわかるように、アドレスデコーダ２６０は、様々なクロック（ＣＬＫ）、制御信号（Ｒ／Ｗ）、タイミング信号（ＢＣＹＳＴ）を受信し、他のゲートアレイ構成要素と同様に、アナログデジタル変換器２３８、フラッシュメモリ２３６、およびメインＣＰＵ２２８上の他の構成要素を選択するためのチップ選択信号を生成する。
【００８０】
ゲートアレイ２２６は、メモリ操作器２６２をさらに含み、メモリ操作器２６２は、アドレスと、ＤＲＡＭ２３０を制御するための他の制御信号とを生成するため、およびＣＰＵ２２８に関連する様々な読み取り／書き込み信号を生成するために、ＣＰＵ２２８アドレスバスに結合される。メモリ操作器２６２は、ＭＡ０１からＭＡ１０のＤＲＡＭアドレス、ＤＲＡＭ２３０に対する行および列アドレス信号、ＤＲＡＭ２３０に対する書込み信号、ならびに入出力読み取りおよび入出力書き込み制御信号を生成するために、ＣＰＵ２２８からのクロック信号、タイミング信号および制御信号に応答する。
【００８１】
ゲートアレイ２２６は、ＣＰＵ２２８の非常に高速のアクセス速度を補償するために、遅延を生成する待機操作器２５４をさらに含む。待機操作器２５４は、（バスサイクル開始信号（ＢＣＹＳＴ）などの）タイミング信号、ＣＰＵ２２８からのクロック信号およびアドレス信号に応答し、適切な遅延期間の後に、実行可能な制御信号をＣＰＵ２２８に入力する。
【００８２】
ゲートアレイ２２６は、識別カードインタフェイス２６４と、プリンタインタフェイス２６６と、キーボード、パッド、およびマウスインタフェイス２６８とをさらに含む。識別カードは、データが読み込まれるＥＰＲＯＭを含む。識別カードインタフェイス２６４は、ＣＰＵ２２８、および識別カード６間の通信をインタフェイスする。プリンタインタフェイス２６６は、プリンタとインタフェイスするために供給されるインタフェイスであり

、入出力レジスタおよび他の従来のプリンタインタフェイス回路を含む。好ましい実施例においては、プリンタインタフェイス２６６は、セントロニクスプリンタとインタフェイスする。キーボード、パッドおよびマウスインタフェイス２６８は、キーボード１８、操作器１２、１４、マウス１６およびＣＰＵ２２８間のデータおよび制御信号の交換を制御する。ダイアラ２６９は、ＣＰＵ２２８のデータバスに結合され、ダイアルパルスを図４に示すモデムボード２２に直接送信する。ダイアラ２６９は、所望する目的地の電話番号を示すデジタル信号を受信し、モデム２２を制御するために、デジタル信号をパルス形式に変換する。

【００８３】

ゲートアレイ２２６は、ＲＧＢインタフェイス２７０をさらに含み、ＲＧＢインタフェイスは、本出願の実施例では、重ね合わせ操作器２１６の重ね合わせ制御回路の一部を含み、重ね合わせ操作器は、図４に示され、図２８で詳述されている。メインＣＰＵ２２８およびＰＰＵ２２４がアクセスする信号のＲＧＢアナログ出力は、ゲームＣＰＵ２００およびＰＰＵ２１２が処理する信号のアナログ出力上に重ね合わされる。ＲＧＢ２７０は、ゲームプレイ関連画面上に重ね合わされたゲームプレイの修正を制御するための編集関連画像セル表示を生成する回路を含む。回路は、ＣＰＵ２２８およびＣＰＵ２００からの重ね合わせ関連制御信号（ＴＯＵＭＥＩ）を受信する。これらの制御信号は、関連するＲＧＢ信号が全てゼロ、すなわち透明である時に、それぞれＣＰＵ２２８および２００を介して出力される。

【００８４】

重ね合わせ操作器２１６は、図２８においてさらに詳細に示される。本出願の実施例では、ＲＧＢ２７０は、図２８に示すように、レジスタＧＳＥＬ３００と、４－１多重装置３０２と、インバータ３０１とを含む。インバータ３０３と、アナログスイッチ３０４および３０６とは、本出願の一実施例においてはゲートアレイ２２６の一部ではなく、図４に示す重ね合わせ操作器２１６内に配置される。図２８に示すように、ＧＳＥＬレジスタ３００は、ＣＰＵ２２８によるプログラム制御下において設定可能である。ＧＳＥＬレジスタ３００ビットのＭ０，Ｍ１は、制御入力として多重装置３０２に結合され、４個の入力信号のうちのどれが多重装置３０２から出力するかを選択する。先に言及したＴＯＵＭＥＩ信号は、空白がある場合、またはＣＰＵ２００およびＣＰＵ２２８のプログラム実行と関連する空の間隔がある場合は何時でも、ＣＰＵ２００およびＣＰＵ２２８によって多重装置３０２へ入力される。ＣＰＵ２２８からのＴＯＵＭＥＩ信号は、反転増幅器３０１を介して多重装置３０２に入力される。

【００８５】

多重装置３０２は、また、「０」および「１」入力信号を受信する。例えば、「０」入力が出力すべき信号としてＧＳＥＬ３００によって選択された場合、インバータ３０３により、アナログスイッチ３０４がつけられ（制御入力において「１」であるため）、アナログスイッチ３０６が消される（制御入力において「０」であるため）。このように、これらの条件下においては、ＣＰＵ２２８からのＲＧＢ信号は、ゲームＣＰＵ２００からのＲＧＢ出力に対抗するＲＧＢ出力として入力される。「１」が選択された多重装置３０２の出力である場合、反対の状況が起こる。そのような操作による視覚効果としては、ＣＰＵ２００によって生成されたゲーム画面とＣＰＵ２２８によって生成されたシステム画面とが、交互に切り替わる。

【００８６】

ＣＰＵ２２８およびＣＰＵ２００がＴＯＵＭＥＩ信号を選択すると、例えばＣＰＵ２２８によって生成されたシステム画面がＣＰＵ２２０によって生成されたゲーム画面上に配置された場合、透過の優先順位を制御する。ゲームＣＰＵ２００に対するＴＯＵＭＥＩ信号が出力された場合、それは、ゲームＣＰＵ信号が透明であり、メインＣＰＵ２２８画面がそれに重ね合わさることを示す。このように、システム画面がゲーム画面に重ね合わさる場合、またはその逆の場合、図２８に示す入力Ｄ２またはＤ３のいずれかが選択される。０または１入力は、ゲーム画面またはシステム画面のどちらが示されているのかを決定す

る。例えば、ＣＰＵ２２８に対するＴＯＵＭＥＩ信号が「０」であって、それが選択された時、インバータ３０１によって多重網３０２の出力は「１」になり、ゲームＣＰＵ２００を選択することになる。そのような信号が出力された場合、ゲームキャラクタが、メインＣＰＵ２２８実行に関連するアイコンのクリアドットを透かして見える。ＣＰＵ２２８によるＧＳＥＬレジスタ３００の設定に従って、画面表示と、ゲーム画面表示およびゲーム編集表示の関連する重ね合わせとが制御される。

【００８７】

図２７に戻って、ゲートアレイ２２６は、また、ゲームＣＰＵインタフェイス２７２を含む。ゲームＣＰＵインタフェイス２７２は、アドレスとＣＰＵ２２８による監視用のデータバスとを介して、ＣＰＵ２００からの幅広いゲーム関連情報を受信するための先に識別したレジスタ複写ＲＡＭを含む。さらに、ゲームＣＰＵインタフェイス２７２は、ゲームデバッギング操作におけるＣＰＵ２２８構内通信を支援するための様々なバッファを含む。ゲームＣＰＵインタフェイス２７２は、また、ＣＰＵ２２８がゲームＣＰＵ２００操作をデバッグすることを可能にする従来のゲームデバッギング操作を遂行するためのアドレスおよびデータトラップ関連回路を含む。ゲームＣＰＵインタフェイス２７２によって、ＣＰＵ２２８は、システムブレイク操作を開始するのに適切な時期を認識するようになる。

【００８８】

ゲートアレイ２２６は、また、水平／垂直（Ｈ／Ｖ）タイマとダウンカウンタ２５２を含み、ダウンカウンタには、ＣＰＵ２２８によって割り込み制御信号を生成して操作器２５０に割り込むための所定の計数を格納しうる。Ｈ／Ｖタイマ２５２は、計数用のタイミングベースとしてＰＰＵ２１２が生成する水平ブランキング（ＨＢＬＡＮＫ）タイミング信号および垂直ブランキング（ＶＢＬＡＮＫ）タイミング信号を使用する。タイミングおよび計数回路２５２を使用することにより、例えば色彩変更のような一定の操作を開始するための特定の水平および垂直表示位置に基づいて、割り込みをすることができる。タイミング回路２５２の出力は、割り込み要求操作器（ＩＲＱＣ）２５０に結合され、割り込み要求操作器２５０は、次に、図２７に示すようにＣＰＵ２２８上の所定のピンに入力される割り込み要求の生成を制御する。

【００８９】

図２７に示すように、ゲートアレイ２２６は、また、通信インタフェイスＲＡＭ２５６を含む。ＲＡＭ２５６は、通信ＲＡＭハンドシェイクポートと、メインＣＰＵ２２８から情報を受信し、ゲームＣＰＵ２００へそのような情報を入力して、ＲＡＭカートリッジに格納するための上述のバッファレジスタろを含む。ＲＡＭ２５６は、ＣＰＵ２２８およびＣＰＵ２００アドレスおよびデータバスに結合される。アドレスデコーダ２５８は、ＣＰＵ２００によるＲＡＭ２５６へのアクセスを制御するためのＣＰＵ２００からのＣＰＵ読み取り信号、書き込み信号およびアドレス信号を受信する。

【００９０】

図２９は、本発明の典型的な実施例で用いられるＲＡＭカートリッジ４の簡略図であり、電源投入中にゲーム処理システムコンソール２からカートリッジを抜いたり、挿入したりすることによって生じる希な状況下で起こりうる潜在的な問題を説明する。図３０で示すように、電源投入中にカートリッジを挿入したり、抜いたりすることによってカートリッジ構成要素が損傷されるという万が一の可能性を防止するために、カートリッジエッジコネクタ内のピンは、修正され、カートリッジＲＡＭメモリを使用可能にするピンは、電力ピンと比較して短くされる。

【００９１】

図３０は、カートリッジ４用のアースコネクタ、アドレスコネクタ、データコネクタ、リセットコネクタおよび電力コネクタを示す。ゲームカートリッジからのリセットピンは、電力回線ピンよりも短いため、信号回線が切断される前にカートリッジが引き抜かれた時に、「リセット」許可信号は、電力が切断されるのに先だってカートリッジＲＡＭに書き込みがなされないようにする。カートリッジが挿入された時、電力回線とデータ回線とは

、リセット信号がハイ状態に置かれるより前に接続され、それにより最後の操作としてＲＡＭ３３６への書込みを可能にする。このようにして、カートリッジ作業用ＲＡＭ３３６に記憶された情報は保護される。カートリッジがコネクタに挿入された場合、カートリッジメモリ回路を使用可能にする前に電力が供給される。カートリッジが抜かれると、短くされた使用可能の信号受信ピンのために、電力が切断される前にカートリッジ回路は使用不可能になる。

【００９２】

図２９に戻って、電源をつけた状態でカートリッジが完全に挿入されると、リセット信号（ＲＥＳ）がハイレベルになる。リセット信号は、ゲート回路３３１を介してデコーダ３３０の可能入力へ入力される。デコーダ３３０は、（図２９に図表で示す）ゲームカートリッジ静的ＲＡＭメモリ３３６のチップ選択ピンへ入力される１つの出力を含む。さらなるデコーダ３３０出力は、４ビットバイナリカウンタ３３２のチップ選択ピンに結合される。バイナリカウンタは、最初０に設定される。デコーダ３３０は、ゲームカートリッジアドレスバスに結合されており、カウンタ３３２への書込み要求を検出し、カウンタ３３２に対して計数を開始させる。書き込みパルスが１５回与えられると、カウンタ３３２の状態は１１１１となり、カウンタ３３２は、ゲート回路３３４に入力される出力信号を生成する。カウンタ３３２の出力がハイレベルにある時にＳＲＡＭ３３６への書き込みが試みられた場合、書き込み信号は、ＲＡＭ３３６（を構成する各ＲＡＭモジュール）へ入力され、それによりカートリッジＲＡＭにデータを書き込むことができるようになる。このように、カートリッジは、電源がついた状態でカートリッジを挿入したり抜いたりする際に構成要素が損傷されたり、データが失われたりする可能性を防止するために、ＲＡＭへのデータの書き込みを行う前に所定のアドレス領域に対して１５回書き込むことを要求する書き込み保護回路を含む。所定のアドレスへ１５回書き込みが試みられると、カートリッジメモリシステムはそのロックが実質上解かれる。実施例のゲーム処理システムにおいて、ＲＡＭカートリッジへの書き込みに費やされる時間は大変少ない。

【００９３】

図２９は、また、カートリッジエッジコネクタ３２９を示す。カートリッジ４に入力される信号には、データバスおよびアドレスバスを通じて搬送されるデータ信号およびアドレス信号がある。さらに、ＣＰＵ読み出し／書き込み制御信号および他の制御信号が受信されるのと同様に、リセット信号が受信される。図２９に示すＲＡＭカートリッジは、機密保護チップ３３３をさらに含み、機密保護チップ３３３は、本件で援用している米国特許番号第４７９９６３５号（対応；特開昭６１－２９６４３３号公報）に記載の方法で、機密保護チップ２２７と共に動作する。ＲＡＭカートリッジ４がＳＮＥＳゲームシステムで使用できるように全てプログラムされた後、機密保護チップ３３３は、カートリッジが、対応する機密保護チップ３３３を含むＳＮＥＳゲームコンソールに関連する真正のゲームカートリッジであるかどうかを決定するための機密保護システムの一部として用いることができる。さらに、ＣＰＵ２２８は、機密保護チップ２２７からの出力を受信し、米国特許番号第４７９９６３５号（対応；特開昭６１－２９６４３３号公報）の特許の教示に従って真実性が決定されない限り、ゲームプログラム実行およびＣＰＵ２００による編集を中止する。この点において、機密保護チップ３３３およびメインＣＰＵ２２８は、米国特許番号第４７９９６３５号（対応；特開昭６１－２９６４３３号公報）に一般的に開示されている処理操作を実行する。そのような処理の結果を示すデータは、安全な方法で交換される。メインＣＰＵ２２８が、ゲートアレイ２２６を介して受信したデータに基づいて、ＲＡＭカセットが本物ではないと決定した場合、ゲームＣＰＵ２００はリセット状態を保持する。

【００９４】

図３１は、図２９に対応するカートリッジ回路の他の実施例である。図２９の同様の回路に対応する図３１の回路は、同様に符号が付けられており、ここでは詳述しない。ＣＬＫ２１Ｍの入力に信号がない場合には、マルチバイブレータは、格納信号をカウンタ３３２に対して生成する。カウンタ３３２は、格納信号によって０にセットされて、ＳＲＡＭへ

の書き込みは禁止状態となる。通常の操作では、ＣＬＫ２１Ｍ信号が連続的にコンソールから提供されて、格納信号が常に禁止状態となり、ＳＲＡＭのアクセスが、図３１の回路と同様に達成される。

【００９５】

図３２は、図３１と共に用いられるカートリッジコネクタの実施例を示す。カートリッジが抜き取られると、ＣＬＫ２１Ｍ信号とリセット信号とが遮断された後に、他の指示信号が遮断される。ＣＬＫ２１Ｍ入力に信号がない場合には、格納パルスが使用可能となる。その後、カウンタ３３２が０データを格納して、ＳＲＡＭへの書き込みは禁止状態となる。図３２では、アドレスバスコネクタおよびデータバスコネクタは、アースコネクタおよび電源コネクタよりも短い。従って、電源が入っている時にカートリッジが押し込まれる際の「ラッチアップ」が防止される。

【００９６】

図３３は、本発明の典型的な実施例に従ったリセット関連回路のブロック図である。システムは、リセットボタン１（３４７）およびリセットボタン２（３４８）、すなわちメインシステムＣＰＵ２２８に関連するリセットボタンおよびゲームＣＰＵ２００に関連するボタンを含む。さらに、図３３は、メインＣＰＵ２２８がソフトウェア制御およびリセット論理２（３４２）を介してゲームＣＰＵ２００をリセットすることができるようにするリセットレジスタ３４４を含む。図３３に示すように、電源３４０が投入されると、リセット論理１が動作してメインＣＰＵ２２８をリセットし、リセットレジスタ３４４をクリアする。メインＣＰＵ２２８は、リセットボタン１（３４７）が押されるのに応答してリセットされてもよい。同様に、ゲームＣＰＵ２００は、リセットボタン２（３４８）が押されることによってリセットされてもよい。代わりに、メインＣＰＵ２２８が、メモリブロック３４６からの命令をアクセスすることにより、リセットレジスタ３４４をセットして、ゲームＣＰＵ２００のリセットを開始する、すなわちメモリ３４５に記憶されたゲームプログラムの初期のような所定の周知の初期状態にゲームＣＰＵ２００を保持するようにしてもよい。図３３は、また、ＣＰＵ２２８およびＣＰＵ２００間の通信が行われるインタフェイスブロック３４９を概略的に示す。他の実施例では、リセットボタン３４８からの信号は、ＣＰＵ２２８の入力ポートに与えられてもよい。リセットボタンが押されると、ＣＰＵ２２８は、リセットレジスタ３４４を介してＳＣＰＵ２００に対してリセット信号を入力する。

【００９７】

図３４は、ユーザがモデルソフトウェアの前景または背景を修正することができるようにするためのシステムブレイクの開始に関連する論理回路のブロック図である。システムブレイクが生成されることになると、メインＣＰＵ２２８は、ラッチを介してフラグフリップフロップ３５０に記憶されるシステムブレイクフラグ信号を生成する。バス状態論理３６０は、従来の比較器回路およびバス状態を識別するための状態表示レジスタを含んでおり、ＣＰＵ２００がプログラムコマンドを読み出すことができることを確認するためにゲームＣＰＵ２００バス状態を検査する。バス状態論理３６０が、ＣＰＵ２００が読み取り／書き込み周期にあると決定すると、バス状態論理３６０は、出力３６１に信号を生成する。ＣＰＵ２００が命令読み取り周期にあるときは、バス状態論理３６０は、コマンドを受信する用意ができていることを示すために、出力３６３上に信号を生成する。

【００９８】

図３４に示すように、アドレスデコーダ３６２は、ゲームＣＰＵ２００システムバスに結合される。アドレスデコーダ３６２は、ゲームＣＰＵ２００が、ＲＡＭカセットデータの特定部分（例えば、ＮＭＩルーチンの最終部）を識別する所定のプログラムアドレス領域にアクセスする時間を検出する。その時に、アドレスデコーダ３６２は、ＡＮＤゲート３６６に入力される出力３６５に信号を生成する。ＡＮＤゲート３６６は、さらに、バス状態論理出力３６１から入力信号を受信する。ＡＮＤゲート３６６の両方の入力がハイレベルにある時は、システムブレイクフラグ信号が、フラグフリップフロップ３５０にセットされる。出力３６１は、ゲームＣＰＵ２００がＲＡＭカセット４からの命令を読み取って

いる時はハイレベルではなく、むしろデータが読み取られているか、またはＲＡＭカセット４に書き込まれている時にアクティブである。

【００９９】

システムブレイクフラグフリップフロップ３５０が設定され、バス状態論理出力３６３がハイレベルにある時は、ＣＰＵ状態は「命令読み取り周期」と示され、許可信号がシステムブレイク論理３５４に入力される。それにより、システムブレイクは、ゲームＣＰＵ２００が命令を読み取ろうとした時点で開始される。この時点において、ＣＰＵ２００は、ゲート回路３５８を介して出力許可信号を生成することによって、プログラムメモリ３５６を可能にしようと試みている。システムブレイク論理３５４は、出力許可信号のプログラムメモリ３５６への結合を不可能にして、命令読み取り周期を中断するのと、プログラムメモリ出力がＣＰＵバスに結合されるのとを防ぐ。

【０１００】

システムブレイク論理は、また、「ＣＯＰ」命令として言及される命令コードを出力する。その後、ＣＯＰコマンドが、ＣＰＵ２００によってプログラムメモリ３５６から読み出されるべき命令と置き換える。システムブレイク論理３５４からの信号に応答するＣＯＰコマンドの後に、モニタＲＯＭ複写保護論理３６８は、許可信号をゲート３５８に対して出力する。このようにして、ＣＯＰコマンドが実行された後に、プログラムメモリ３５６内のモニタＲＯＭプログラムは、ＣＰＵ２００によってアクセス可能となる。ＣＯＰコマンドがＣＰＵ２００によって受信されると、ＣＰＵは、モニタＲＯＭ２０４からプログラム命令を実行する。このように、ＣＰＵ２００は、ゲームプログラムの実行を停止し、ＣＯＰコマンドに応答してモニタＲＯＭ２０４からの命令の実行を開始し、それによってゲーム編集および作成機能の遂行のために制御のメインＣＰＵ２２８への切り替えが開始する。

【０１０１】

さらに、モニタＲＯＭ複写保護論理３６８は、ゲームプログラムが実行中に、プログラムメモリ３５６に格納されているモニタＲＯＭプログラムからの不所望な読み取りを防止する。従って、モニタＲＯＭプログラムを複写するためにユーザのプログラムから故意にアクセスすることが不可能となる。モニタＲＯＭ複写保護論理を用いた図３４に示す回路は、モニタＲＯＭプログラム複写保護機構として有益である。

【０１０２】

システムソフトウェアの詳細な説明

次にシステムソフトウェアのより詳細な説明へ移って、そのようなソフトウェアは、図解実施例中で「マリオファクトリー」と呼ばれるゲームプログラム編集ツールを含む。マリオファクトリーは、不慣れなユーザがモデルソフトウェアを用いてビデオゲームを容易に作成するのを支援するために、主に設計されている。

【０１０３】

図３５は、本発明の実施例に関連して、電源投入から「マリオファクトリー」ユーティリティタイトル画面が出現するまでの処理のシーケンスを示すフローチャートである。電源を入れると（３８５）、最初の接続画面が表示される（３８６）。その後、著作権情報画面が生成される（３８７）。その後、実施例に関連するゲーム編集ツールが、図４に示すフラッシュメモリ２３６内に既に記憶されているかどうかを決定するためのチェックが行われる（３９１）。ブロック３９１でのチェックにより、ツールがフラッシュメモリ２３６内にあることが示されると、ツールはＣＰＵ２２８に関連するＤＲＡＭ２３０に転送され、マリオファクトリーの実行が開始できるようになる。従って、編集ツールがフラッシュメモリ内に既にあれば、マリオファクトリーは自動的に格納される。ブロック３９１でのチェックにより、編集ツールがフラッシュメモリ２３６内にないことが示されると、３８８において、ツールディスケットがフロッピーディスクドライブ内にあるかどうかを決定するためのチェックが行われる。フロッピーディスクディスケットがフロッピーディスクドライブ１９９内にない場合には、「ツールディスクを挿入して下さい」というメッセージがユーザに対して表示される（３８９）。ツールディスクが挿入されていることが検

出されると、ツール導入画面が生成され（３９０）、マリオファクトリーの格納が完了する。

【０１０４】

本発明の一実施例に従って、モデルソフトウェアは、２つの異なるファイルが記憶されるように格納される。２つの異なるファイルとは、ゲームの修正不可能な面を含むベースファイルおよびユーザが修正しうるゲームの諸要素を含むユーザファイルである。代わりに、ベースファイルのみが最初に格納されて、ゲームプレイの基本的な面のみをユーザに提供して、新規のゲームをベースファイルに関連するゲームのジャンル内に構築および作成するようにしてもよい。

【０１０５】

ブロック３９２において、モデルソフトウェアがフロッピーディスクドライブにあるかどうかを決定するチェックが行われる。モデルソフトウェアがフロッピーディスクドライブ１９９内にない場合は、ユーザは、モデルソフトウェアディスクをインストールするように促され（３９４）、モデルソフトウェア格納画面が生成される（３９３）。その後、ユーザファイルがユーザによって選択されているかどうかのチェックが行われる（３９５）。ユーザファイルは、ベースファイルとは異なるディスクに格納されていてもよい（このため、ブロック３９５におけるユーザファイルの選択には、別のディスクの挿入を必要とする場合がある）。ユーザファイルは、モデルソフトウェアビデオゲームに対してこれまでなされた変更を含む。最初は、ユーザファイルは、モデルソフトウェアが初期ゲームプレイを定義しているデフォルトステージ内にある。ユーザがゲームを変更すると、デフォルト情報が変更されて、ユーザの編集上の修正が反映される。ゲームカートリッジおよびフロッピーディスクの両方は、ベースファイルとユーザファイルとを含むが、フロッピーディスクは、ゲームカセットが記憶しないようなゲームの元のデフォルトバージョンを記憶する。ＲＡＭカセットデータは、デフォルトデータがカセット４上に常駐しないようにリアルタイムで修正される。ユーザファイルが選択されると、ユーザファイル格納画面が表示される（３９８）。ユーザファイルが選択されていない場合には、ユーザは、ユーザディスクを挿入するように促され（３９６）、ユーザは、ユーザファイルを選択する新たな機会を持つようになる（３９７）。その後、ユーティリティタイトル画面が図８に示すように表示される（３９９）。

【０１０６】

ブロック３９２において、モデルソフトウェアは、別のディスクを挿入することによって、ユーザの所望通りに変更することができる。ユーティリティタイトル画面が表示されると、ユーザは、ユーザが他のユーザファイルを選択することができるブロック３９５に分岐して戻るような特定のファイルアイコンをクリックしてもよい。代わりに、ルーチンは分岐して、モデルソフトウェアを変更することによってすべてのジャンルのゲームを変更しうるブロック３９２に戻ってもよい。マリオファクトリープログラムは、多様な編集ツールを含み、例えば特定のキャラクタを描画するための「ドット」や、動画を制御するための「動画データ」等がある。

【０１０７】

ゲームプロセッサシステムは、本明細書で説明する実施例に従えば、ＣＰＵ２００の制御下のソフトウェアと、メインＣＰＵ２２８の制御下のソフトウェアとを含む。ＣＰＵ２００によって処理されるソフトウェアは、上述のモデルゲームソフトウェアと、ユーザが作成したゲームプログラムとを含む。メインＣＰＵ２２８は、種々のユーティリティプログラム、オペレーティングシステム、周辺ドライバプログラムならびにＢＩＯＳおよびＩＰＬソフトウェアを実行する。メインＣＰＵ２２８によって処理されるユーティリティソフトウェアは、ゲーム編集ツール、ネットワークソフトウェア、文書処理ソフトウェア、ディスク管理ソフトウェア等を含む。オペレーティングシステムおよび周辺ドライバソフトウェアは、以下に述べるＢＩＯＳソフトウェアによっては充分に支援されていない周辺装置を支援するためのサブルーチンを含む。

【０１０８】

ＢＩＯＳソフトウェアは、上述のシステムハードウェアと直接対話を行うための低レベルルーチンを含む。より特定的には、システムのメインＣＰＵ２２８部と、ゲームＣＰＵ２００のサブシステムとの間の情報交換は、ソフトウェアから捉えると、ＢＩＯＳ　ＲＯＭ２３４内に常駐するルーチンと、モニタＲＯＭ２０４内に常駐するルーチンという２つのＢＩＯＳオペレーティングシステムルーチンによって制御される。オペレーティングシステムルーチンからのそのような情報転送の間に遂行される処理を、以下に説明する。ＢＩＯＳルーチンは、図３および図４に示すマウス１６，キーボード１８，操作器１２および１４，表示装置１５およびモデム２２のような種々の入出力装置を制御し、処理装置の構内通信、メモリ保守機能およびＰＰＵ制御機能を制御する。ＣＰＵ２００　ＢＩＯＳソフトウェアによって制御されるＰＰＵ２１２の機能は、例えばＰＰＵのカラー生成ＲＡＭが所定の所望するカラーマップを定義するようセットすること、ビデオＲＡＭ２１４に対してデータを転送すること、重ね合わせ優先順位をセットすること等を含む。制御される画面表示関連機能は、カーソル位置をセットすることを含む。制御されるキーボード機能は、キーボードの初期化、キーボードバッファデータの取得およびキーボードステータス情報の取得を含む。同様に、マウスに関しては、ＢＩＯＳソフトウェアによって制御される種々の機能は、マウスの初期化や、マウスのカーソル位置の取得等を含む。ゲームＣＰＵ２００の操作の制御に関連するＢＩＯＳソフトウェアは、ゲームＣＰＵの初期化、音声処理装置２０８および作業用ＲＡＭ２１０に対するデータの送信、ならびにメインＣＰＵ２２８とのデータの交換に関連する機能を含む。メインＣＰＵ２２８に関連するＢＩＯＳソフトウェアは、リアルタイムクロックのセット、ゲートアレイ２２６内のタイムアウトカウンタのチェックおよびＭＳ　ＤＯＳのような従来のオペレーティングシステムにおいて通常遂行されるようなオペレーティングシステムハウスキーピング機能の遂行のような制御機能を含む。オペレーティングシステムソフトウェアは、図４のＳＣＳＩボード２４を介してＩＢＭ互換ＰＣから受信されたゲームプログラムを実行および編集できるようにするために、他のコンピュータからいくつかのコマンドを受信することができる。

【０１０９】

ＩＰＬソフトウェアは、初期プログラム格納を行う役割を果たしており、システム立ち上げ時にゲームプロセッサシステムステータスをチェックして、オペレーティングシステムルーチン内のＢＩＯＳを立ち上げる。初期プログラム格納（ＩＰＬ）ルーチンは、システムの開始時に実行され、まず開始メッセージが表示されることになる。開始メッセージは、図３５に関連して上述したような著作権警告を含む。その後、ＩＰＬは、会員識別カード内の巡回冗長コード（ＣＲＣ）をチェックする識別コードチェックを遂行する。識別カードが権限を有していると決定されれば、システムが立ち上がる。識別チェックによって、識別カードが権限を有していないことがわかると、権限を有しない旨を示す表示が生成される。識別コードチェックによってシステムが立ち上がることができるようになると、ハードウェアステータスチェックが行われる。ハードウェアステータスチェックは、現在のシステム構成のＲＡＭのサイズを決定し、接続されたＲＡＭが読み取りおよび書き込みが可能である旨を確認するチェックを行う。その後、フロッピーディスクドライバ、モデム、プリンタ、ハードディスクならびに制御ユニットおよび拡張ユニット内の他の装置が適切に接続されているかを確認するためのチェックが行われる。すべてのハードウェアステータスチェックによって適切な接続が確認もされない場合には、システム障害を示す適切な表示が生成される。

【０１１０】

すべてのハードウェアステータスチェックによって適切な接続が示されると、その後、オペレーティングシステム識別コードのチェックが行われる。このチェックでは、オペレーティングシステム内の識別コードがチェックされて、チェックが通れば立ち上がり処理が行われる。チェックに失敗すると、システム障害を示す表示が生成される。識別コードチェックによって適切な識別コードが確認されると、オペレーティングシステムが立ち上がる。

【０１１１】

機密保護チェックは、システム全体に渡る多重レベルにおいて機密保護システムが導入されるように設計される。上述したように、初期プログラム格納（ＩＰＬ）において、権限のない識別カードを検出するための識別コードチェックが行われる。識別コードチェックは、権限のないＯＳ／ＢＩＯＳが使用されている旨を示すためにも行われる。さらに、上述したように、機密保護チップがＲＡＭカセット４内で用いられて、システムが権限のないハードウェアと結合されているかどうかの決定を行う。さらに、ＢＩＯＳ／ＯＳレベルにおいて、ベースファイルを不適切にアクセスしているかどうかを決定するための識別コードチェックが行われる。識別コードチェックは、権限のないネットワークアクセスおよび権限のないユーティリティの使用があるかどうかをも決定する。このような識別コードチェックに失敗すると、行おうとしたファイル操作は不可能となる。ＢＩＯＳ／ＯＳは、上述した以外のマシンアーキテクチャによる読み出しを困難にするような関連する特有データフォーマッティングおよび／またはファイル圧縮方法をも用いる。

【０１１２】

オペレーティングシステムは、コマンドの解釈、メモリの管理、一時的なオペレーティングシステム部内の読み取りおよびコマンドルーチンの開始を行う中核部を含む。中核部は、識別カードからの「構成ファイル」と呼ばれるファイル内を読み取る。「構成ファイル」とは、周辺ドライバのセットアップのような事項を記録して、一時的なオペレーティングシステム、一時的なＢＩＯＳ、周辺ドライバ等をメモリ上にアンパックして、それらを管理するファイルである。オペレーティングシステムのコマンドルーチン部は、中核部からの命令に基づいた実際の処理を遂行するサブルーチンを含み、オペレーティングシステムの周辺ドライバ部は、上述したような種々の周辺装置へのアクセスを取り扱うサブルーチンを含む。

【０１１３】

ファイルは、通常６４バイトのヘッダファイルを含んで編成され、ヘッダファイルにはファイル名、日付、ファイルサイズ、ファイル属性、コード作成器、識別コード、暗証等のようなフィールドを含む。ファイル作成者については、ベースファイル内にのみ記録され、購入者の識別コードは、購入時に記録される。ファイル属性フィールドは、識別コードが作成者コードと一致する場合にのみファイルを可能にするか、または、代わりに、暗証が一致する場合にのみファイルを可能にするビットを含む。従って、ファイル属性フィールドまたは他のビットの状態によっては、ファイルを可能にする場合、不可能にする場合またはある機密保護チェックに通過するかどうかに基づいて可能にする場合があり得る。ファイル属性情報は、ベースファイルアクセスの種々の条件をセットでき、例えば、アクセスが購入者によってなされれば書き込みは禁止であるとか、アクセスが購入者によってなされなければ、読み出しおよび書き込みは禁止でかつファイル名は表示されないようにできる。オペレーティングシステムは、ベースファイルアクセスコマンド（例えば格納、記憶、認証、複写等）、ファイル管理コマンド、ネットワークコマンド、暗証コマンドおよびＲＡＭカートリッジコマンドを含む。システムは、暗証を構成変数としてセットする暗証コマンドを用いる。構成変数は、ヘッダファイル内の暗証と比較される。構成変数がセットされていない場合は、暗証を有していないとみなされる。

【０１１４】

上述のように、モデルゲームソフトウェアは、ゲーム作成を支援し、ゲームＣＰＵ２００によって実行されるソフトウェアである。このソフトウェアは、独自のゲームが簡単なゲーム作成ツール操作によって作成可能であるように設計されている。本発明の好ましい実施例では、モデルソフトウェアは、「射撃ゲーム」、ロールプレイゲーム等のような種々のゲームのジャンルに従って販売される。本明細書では、ビデオゲームプログラム設計および作成の手ほどきを受けていない者によるゲーム作成のために主に設計された「マリオファクトリー」というゲーム作成ツールの例の詳細を説明する。しかしながら、本発明は、中級および上級ユーザが本明細書で述べるよりもより複雑な制作さえも行うことができるようなゲーム作成ツールの使用も考慮している。しかしながら、現時点での好ましい実施例は、マリオファクトリーで説明される。モデルソフトウェアは、ベースファイルおよ

びユーザファイルを含む。ベースファイルは、ユーザが変更することのできないファイルであり、ユーザが変更不可能なモデルソフトウェアおよびそれに関連する独自のゲームの部分を含む。ユーザファイルは、ユーザが変更することのできるファイルである。ユーザファイルは、ユーザが修正してユーザ独自のプログラムおよびユーザ独自のデータを作成することができるデフォルトプログラムおよびデフォルトデータを含む。ベースファイルは、複写保護されており、ベースファイルの複写を取得するにはモデルソフトウェアを購入するしか方法がないようになっている。ユーザファイルは、複写保護はされておらず、例えばユーザファイルをネットワーク上にアップロードすることによって自由に取得することができる。ユーザの作成した独自のゲームは、他のユーザに対してユーザファイルを送信することによって送信されてもよい。

【０１１５】

マリオファクトリーでは、ゲームは、多くの構成「ユニット」を処理することによって処理される。例えば、図３６では、スーパーマリオ画面は、プレイヤによって制御される（例えば、ゲーム操作器１２および１４によって制御される）ユニットであるマリオと識別されているユニット１と、「敵」ユニットと名付けられているユニット２と、音楽ユニットであるユニット３とともに示される。グラフィックデータやキーに対する適切な応答および効果音といったユニットに関連するすべてのデータおよびアルゴリズムは、以下で詳しく述べるように、ユニット用に記憶される。このシステムは、グラフィックデータに関連しない音楽ユニットに対する配慮もなされている。ゲームはツールアイコンを選択して変更を行うことによって編集される。例えば、図３６で、ユニット１が選択されると、マリオグラフィック、音声または条件に対するプログラムされた応答についてツール選択が行われる。プログラマから見れば、ユニットは、ゲームプレイの特徴およびゲームＣＰＵ２００側に割り当てられた幅広い範囲の他のデータを記憶するためのメモリスペースとみなされうる。

【０１１６】

前述したように、本発明の好ましい実施例は、ピクチャ、音声およびプログラムを変更するための幅広い種々の編集ツールを含む。特定のゲーム専用のエディタが、モデルソフトウェア内に拡張ソフトウェアとして含まれてもよい。例えば、射撃ゲーム専用の専門のマップエディタまたは敵パスエディタが用いられてもよい。同様に、ロールプレイゲーム専用のマップエディタ、対話エディタおよびイベントエディタが含まれてもよく、パズルゲーム内にルールエディタが拡張ソフトウェアとして含まれてもよい。

【０１１７】

このような種々の型の各ツールは、ユーザが選択できるように表示される。アイコンは、層状に編成され、特定のときに必要とされるアイコンのみが、ユーザに対して表示される。必要でないアイコンは表示されないが、より正確に言えば、必要でないアイコンは、モデルソフトウェア内に含まれているステータスファイル内に記録される。

【０１１８】

ゲーム編集は、編集ユニットおよび関連データ構造を循環する。本発明の実施例では、異なる編集機能が、システムブレイク時にアイコンをクリックするか、またはツールボックス画面からアイコンをクリックすることによって制御される。別の実施例においては、左マウスボタンまたは右マウスボタンのどちらが押圧されたかに基づいて異なる編集機能が制御される。ゲームプレイが開始された後、システムブレイクが開始し、マウスカーソルが画面上に現れる。画面上に表示された特定のユニットの編集を所望する場合には、例えば左マウスボタンが例えば図３６に示すユニットのうちの１つに対してクリックされ、選択されたユニットを編集するための全ツールが表示される。全ツールには、グラフィック編集機能、プログラム編集機能および音声編集機能が含まれる。右マウスボタンがクリックされると、例えば「取り消し」「ゲーム」「ファイル」「ツールリスト」「ユニット」および「終了」のような種々のオプションが表示される。ツールリストオプションが選択されると、ツールは、例えばツリーチャートとして示され、選択されたツールで編集可能なユニットが制御されて点滅する。

【０１１９】
この操作中に、編集画面が、ゲーム画面上に重ね合わされる。所望する編集が行われた後、ゲームプレイが再開されてもよい。左マウスボタン編集操作では、ゲーム画面が表示されてシステムブレイクが開始した後に、マウスカーソルが画面上に現れ、ユーザは、編集しようとするキャラクタをクリックする。編集しようとするキャラクタは、識別されたユニットとともにボックス内に現れる。その後、ユニットの修正が、種々の編集ツールを介して生じる。編集後、ゲームプレイが再開されてもよい。
【０１２０】
右マウスボタン編集操作に関しては、結合された画面上においてツールリストが生成される。その後、ユーザはツールを選択し、編集可能な各ユニットがそのツール用に表示される。その後、マウスを介して、別のツールが選択されてもよく、異なる編集可能な「ユニット」が表示される。キャラクタがクリックされると、当該キャラクタがボックス内に現れ、メニューが現れて、左マウスボタン編集におけるのと同様の方法で編集が続く。右マウスボタン編集において、マリオファクトリー操作中に、ステータスファイルがチェックされて、ステータスが表示される。ユニットヘッダ（ユニットを編集できるツールのような情報を記憶するユニットの先頭に置かれるデータ領域）がチェックされて、編集可能なユニットが検索される。
【０１２１】
本システムにおいては、モデルソフトウェア選択画面があるので、たとえモデルゲームソフトウェア自体がなくても、マリオファクトリー作成ツールを開始することができ、基本ツールを用いることができる。マリオファクトリープログラムは、ステータスファイルを含み、ステータスファイルは、メモリ内に位置付けられるべきツールのラベル名を記録するテキストファイルである。ステータスファイルをアクセスすることによって、マリオファクトリープログラムは、どのツールがＤＲＡＭ２３０に記憶される必要があるかを決定することができる。ＤＲＡＭ２３０内に記憶されるツールの中には、アイコンとして表示されるものもある。そのようなツールは、基本ツールと、メインＣＰＵ２２８コード内に書き込まれたオブジェクトファイルから選ばれた拡張ツールとを含んでもよい。
【０１２２】
図３７は、プログラミングが「ユニット」を基本としてどのように構築されているかを示すブロック図である。ゲームＣＰＵ２００のメモリアドレススペース４０１内には、編集されうる各ユニットを一意に識別するユニット関連データが記憶される。編集されうるユニットと、ゲームＣＰＵメモリアドレススペース内に記憶されたユニット関連データ構造との間には、一対一対応がある。
【０１２３】
図３７に示すように、ユーザが、表示画面１５上に示されたユニット３を、例えばシステムブレイク処理中に選択すると、メモリスペース４０１の対応部は、ユニットヘッダと、データテーブルと、ユニット３関連プログラムとを記憶する（またはポイントする）ユニット３に関連するユニットデータ構造（またはポインタ）を含む。ユニットヘッダは、特定のユニットを編集するために用いることができるツール用のラベルと、編集されたデータ用の記憶アドレスとを含む。ユニットヘッダのグループも、メインＣＰＵ２２８アドレススペース内に記憶される。データテーブルは、以下に述べる幅広い範囲のゲームプレイ／オブジェクト特徴情報を含み、その情報には、グラフィックおよび音声だけではなく、各キャラクタのパス動作を示す情報も含まれる。本質的に、各ユニットに関連するグラフィック、音声およびプログラムは、各ユニットに保持されまたは貼り付けられているとみなされうる。
【０１２４】
メインゲーム処理ルーチンは、ユニット１からＮをそれぞれ処理するための命令のシーケンスを含む。メインルーチン（４０５）は、各ユニットプログラムをアクセスするためのサブルーチンへジャンプ（ＪＳＲ）のシーケンスを含む。各ユニットプログラムは、いかなるフレーム中においてもメインルーチンから呼び出されうるという仮定の下に設計され

た命令を含む。メインルーチン４０５は、メインＣＰＵ通信ルーチンへのジャンプをも含む。この通信ルーチンは、マウスクリックポイントのメインＣＰＵに対する信号伝送、編集ツールに関連するデータ編集に関連する信号の受信およびメインＣＰＵシステムブレイク信号のチェックといった処理を扱う。

【０１２５】

図３８から図４０は、ユニットデータの処理と、ピクチャ信号および音声信号のユーザの表示画面に対する出力とに含まれる処理のシーケンスを示すフローチャートである。図３８に示すように、ユニット処理は、ユニット画面エディタによるユニットの処置を示すユニットマップ画面をシステムが参照して開始する。例えば、キー制御によって背景画面がスクロールされると、ユニット画面マップ上の新規の部分が参照されて、新規のユニットが現れるかどうかを判断する（４０５）。初期化処理を行って、ユニットマップ画面上に新規ユニットが見つかった後、エントリが稼働中ユニットリスト内に作成されて、ユニットアクセスカウンタが、ユニット０にセットされ（４０７）、ルーチンは、ユニット処理ループへ入って、ユニットオフセットアドレスが計算される（４０８）。ブロック４０９に示すように、ユニットに関連するユニットプログラムが実行され、該当する場合には、動画処理（４１０）および音声処理（４１１）が遂行される。その後、アクセスされたユニットを常に把握しているカウンタが増分され（４１２）、最終ユニット番号がアクセスされたかどうかを決定するためのチェックが行われる（４１３）。ブロック４１３のチェックによって最終ユニット番号がアクセスされていないことが示されると、ルーチンは分岐してブロック４０８に戻り、処理される次のユニット番号が、オフセットアドレス計算によってメモリ内に位置付けられる。

【０１２６】

図３９および図４０は、ブロック４１０の動画処理をさらに詳しく示す。図３９に示すように、動画処理は、処理されているユニットのために表示されるポーズ番号を識別することによって開始する（４１４）。その後、キャラクタアドレスが計算され（４２４）、計算されたアドレスに記憶されたキャラクタデータは、処理のために、メモリバッファ領域に送信される（４２５）。その後、ブロック４２６において、ユーザデータ置換フラグがセットされているかどうかを決定するチェックが行われる。ユーザ置換がセットされていれば、ユーザ選択データが置換される（４２７）。ユーザ置換フラグがセットされていなければ、ブロック４２５で識別されたバッファ領域内に記憶されたキャラクタデータは、オブジェクト属性メモリ領域へ送信される（４２８）。

【０１２７】

図４０は、図３９のブロック４１４の処理のための表示ポーズ番号の決定に関連する処理ステップを示す。最初に、ブロック４１５において、表示アニメ番号と、図４４に示すような「ユニット」とともに記憶された所定のバックアップデータとを比較するチェックが行われる。一致する場合には、ルーチンは分岐してブロック４１７へ進む。一致しない場合には、アニメデータは、図４４で詳細に示すオブジェクトユニットＲＡＭへ送信される。その後、「テンポ」データが、フレームカウンタの内容から除かれる（４１７）。フレームカウンタが０以下かどうかを決定するチェックが行われる（４１８）。フレームカウンタが０以下の場合は、ポーズ番号が増分される（４１９）。ポーズ番号がポーズ番号の許容上限を超えているかどうかを決定するチェックが行われる（４２０）。ポーズ番号が上限を超えている場合は、動画シーケンスにおいてユニットポーズを決定するポーズ番号は、例えば図１４の動画ツールを用いて定義されるポーズ上限事後番号と同値にセットされる（４２１）。その後、ポーズデータは、図３９のブロック４２５に関して識別されたバッファ領域に送信され、ポーズポイント位置データ（ｘ，ｙ）が、バッファに送信される（４２３）。フレームカウンタが０以下でない場合は、ルーチンは分岐して直接ブロック４２３へ進む。

【０１２８】

図４１は、処理されるユニットに関連する音声処理のフローチャートである。最初に、ブロック４２９において、ステータスレジスタと、図４４のオブジェクトユニットＲＡＭ内

に記憶されたステータスレジスタバックアップとを比較するチェックが行われる。４２９でのチェックで、一致していないことが示されると、効果音データオフセットアドレス４３０を決定するための計算が行われる。効果音番号は、計算された効果音オフセットアドレスに追加されたユニットエントリアドレスに基づいて計算される（４３１）。効果音番号は、その後、音声処理装置２０８ポートへ送信される（４３２）。実効果音データは、すべての効果音番号とともに作業用ＲＡＭ２１０に記憶される。音声処理装置は、効果音番号および実効果音データとともに音声を作成する。ブロック４２９でのチェックで、ステータスレジスタ内容がステータスレジスタバックアップ内容と一致していることが示されると、音声処理サブルーチンを出て、ルーチンは図３８のブロック４１２の処理へと続く。

【０１２９】

図４２および図４３は、第１および第２のモデルソフトウェアフォーマットを図式的に示しており、これらのフォーマットは、メインＣＰＵ２２８が情報を効果的に処理して容易に編集できるように構成されている。図４２および図４３の情報の多くについては、本明細書では説明しないが、本明細書の説明に照らして、当業者によって理解されるであろう。図４２に示すように、第１のモデルソフトウェアフォーマットは、背景関連であり、メモリ内の位置のセットに記憶されたステージ（背景）データをポイントする「ステージ」ポインタセクションを含む。次に、ステージデータは、幅広い種々の追加情報をポイントし、追加情報には、例えばフォントデータ、カラーパレットデータ、他の背景関連情報データがあり、他の背景関連データには、例えばマップデータ、ユニットマップデータ、パネルマップデータ、パネルユニットマップデータおよびパネルデータがある。

【０１３０】

図４３は、第２の移動オブジェクト関連モデルソフトウェアフォーマットを示しており、そのフォーマットでは、マリオのような識別「オブジェクトユニット」が、関連するオブジェクトユニットメモリ区分に割り当てられる。各識別オブジェクトユニットに関連するのは、オブジェクト動画（アニメ）ポインタで、このポインタは、図４３に示すように、動画データのブロックをポイントする。さらに、各オブジェクト関連ユニットは、各オブジェクトユニットに関連した種々のプログラムを識別するオブジェクトプログラムポインタ情報を含む。

【０１３１】

図４４は、オブジェクトユニットＲＡＭ情報領域およびゲーム背景関連ＲＡＭ領域を含むＲＡＭ情報内容のメモリマップを図式的に示す。図４４に示すように、オブジェクトユニット情報は、オブジェクトユニット識別と、ステータスレジスタおよび座標位置と、キャラクタサイズやポーズ情報等のような幅広い範囲の他のオブジェクト情報とを含む。この情報は、図３に示すメインＣＰＵ２２８およびＤＲＡＭ２３０のバッファ領域に記憶されており、制御ファイルを用いて編集するために用いられる。制御ファイルは、編集を支援するためのいくつかの相互参照データ、ラベルテキストおよびメッセージテキストを有する。制御ファイルに記憶されたデータは、ゲームプログラムの実行には必要ないが、編集をより容易にするために用いられる。ゲームプレイ中に用いられる背景音楽データは、メインＣＰＵ２２８システム部からゲームＣＰＵ２００セクションに戻って送信される。

【０１３２】

図４５は、メインＣＰＵ２２８およびゲームＣＰＵ２００メモリマップを示しており、データが抽出されて、システムブレイク中にゲームＣＰＵ２００からメインＣＰＵ２２８へ複写される方法を示している。図４５は、また、編集中に、メインＣＰＵ２２８とゲームＣＰＵ２００とが効率的に相互動作にできるように、ある共通値が各メモリ領域にどのように保持されるかを示す。

【０１３３】

図４５のＣＰＵ２００部内のデータ領域には、グラフィックデータ、音声データおよびステータス（処理プログラム）が、動画データ、ＡＳＭデータおよびＢＡＳＩＣデータとして記憶される。また、ゲームＣＰＵ２００のメモリ領域は、図３７に関連して説明したメ

インプログラムおよびＳＨＶＣ－ＢＩＯＳ内の関連メインプロセッサ通信ルーチンを含む。複写保護機構を備えた編集不可能なベースファイルは、データ領域、ユニットポインタおよびアセンブラメインプログラムの一部として記憶される。メインＣＰＵ２２８メモリシステムには、マリオファクトリーの基本編集ツールに加えて、ＣＰＵコードが記憶される。ＣＰＵ２２８のメモリ領域は、また、オペレーティングシステムプログラムおよびメインＣＰＵ通信ルーチンを記憶するためのメインプログラム領域を含む。さらに、ベースファイルは、制御ファイルとして、かつユニットデータ、ユニットポインタおよび編集用のいくつかのマイナデータを含むバッファ領域内の一部として記憶される。これらの要素は、ユーザが選択したモデルソフトウェアによって制御される格納プログラムを介して、マリオファクトリープログラムに格納される。

【０１３４】

マリオファクトリーが実行されると、図４５に示すようにメインＣＰＵ２２８メモリに割り当てが行われて、タイトル画面が表示される。次に、格納アイコンがクリックされると、モデルソフトウェアのいくつかのファイルはＣＰＵ２２８メモリに格納され、いくつかのファイルは各メモリ内の通信ルーチンを介してゲームＣＰＵメモリへ格納される。モデルソフトウェアは、ＳＣＰＵ２００メモリ内に、データ領域に書き込まれたユニット情報およびユニットポインタを含む。モデルソフトウェアは、ＣＰＵ２２８内の制御ファイル領域に記憶される相互参照データ、ラベルテキストまたはメッセージテキストとして編集するためのマイナデータである制御ファイルデータをも含む。制御ファイルデータは、編集されたデータが、（編集を禁止する）ベースファイルデータであるか、または（編集を可能にする）ユーザファイルデータであるかを示す属性データを含む。さらに、制御ファイルデータは、編集の限度示す限度データを含む。例えば、マリオの速度が８ドット当たり１フレームを超えることができないように指定した限度がセットされてもよい。本発明の実施例に従えば、ユーザファイルは、ベースファイルのユニット情報を除くデータ領域データおよびユニットポインタデータを含む。ユーザデータは、また、デフォルトデータと編集されたデータとの差を含んでもよい。

【０１３５】

図４６のフローチャートは、ゲームＣＰＵ２００およびメインＣＰＵ２２８がゲーム作成処理中に遂行する処理のシーケンスを示す。このフローチャートは、ゲームの修正を行うためにゲームプログラムがどのように実行、停止され、その後、編集が生じた点からどのように再開されるかを示す。フローチャートの左部分は、ゲームＣＰＵ２００のフローを示しており、フローチャートの右部分はメインＣＰＵ２２８のフローを示す。メインＣＰＵ２２８のフローは、対話およびデータ交換を示すためにゲームＣＰＵのフローと並行して示される。そのような相互交換はゲーム画面を停止することと、例えばマリオを編集するための特定のユニットを選択することとによって生じる。列間の矢印は、２つのプロセッサ間でのデータ交換を示す。

【０１３６】

まず、ゲームプレイが開始されることになると、ゲームＣＰＵ２００が制御され、前述したようなモデルソフトウェアで実施されたゲームをプレイするためにメインルーチンの実行を開始する（４３３）。システムブレイクが開始すると、前述したようにＣＯＰコマンドが生成され、ベクトルジャンプを生じて、ソフトウェアがモニタＲＯＭ２０４の中から実行されることとなる。システムブレイクがなされて、例えばマリオのキャラクタが修正用に選択されると、ユニット識別子が生成される。システムブレイク中に変更される情報は、ＣＰＵ作業用ＲＡＭ２０２内に記憶される。ベクトルジャンプの後、ゲームＣＰＵが、モニタＲＯＭ２０４から実行される。

【０１３７】

図４６で示すように、メインＣＰＵ２２８は、後述のプログラムループを実行して、モデルソフトウェアゲームプレイにおいて変更を行うためのユーザからの停止要求を待つ。マウスボタンがユーザによって押圧されると、停止要求が検出されて（４４０）、ソフトウェア割り込みが生じ、ＣＯＰコマンドがゲームＣＰＵに入力される（４４１）。

【０１３８】
画面表示作成中にシステムブレイクが行われると、生成された表示は損なわれる。図３４に示すＳＣＰＵブレイク論理は、例えば垂直ブランキング期間中に生じるマスク不可能割り込み（ＮＭＩ）の終了時などに生成された表示に対する影響が不可能になるまで、システムブレイクを遅延させる。メインＣＰＵ２２８が、ＮＭＩ期間の終了時に、ＳＣＰＵブレイク論理によってＣＰＵ２００にＣＯＰコマンドを提供すると、ＣＰＵ２００はモニタＲＯＭ２０４から実行を開始して、変更するデータをメインＣＰＵ２２８バッファ領域に送信する。メインＣＰＵ２２８は、変更するキャラクタの座標位置、オブジェクトサイズおよび識別ユニット番号を知らされる。ＣＰＵ２２８のユニット作業用ＲＡＭ領域は、オブジェクトに関連するステータス情報を含む。その後、ゲームＣＰＵは、モニタＲＯＭ２０４の中から実行を開始することによって応答する。

【０１３９】
ブロック４３４で示すように、ゲームＣＰＵ２００は、選択されたユニット作業をメインＣＰＵ２２８に送信し、メインＣＰＵ２２８はユニット作業を受信する（４４２）。図４５に示すメモリマップでわかるように、ユニット作業は、ＣＰＵ２００メモリスペース内に含まれ、識別されたユニット作業のコピーが、メインＣＰＵ２２８ユニット作業複写領域に送信される。図４４は、オブジェクトユニットに関連する記憶された情報を図式的に示しており、図４７は、稼働中ユニットのためのユニットポインタ領域と、ユニット識別番号、表示ポイント、キャラクタサイズ、ユニット内のプログラムカウンタ等の識別された稼働中ユニットのための代表的なユニット情報とを示す。図４８は、全ユニットのためのユニットポインタ領域を示しており、ユニットポインタ領域は、動画データポインタ、プログラムポインタ、音声ポインタ等のようなユニットポインタデータを識別する。好ましい実施例では、図４４の右列に示す全ユニット情報は、作業複写領域に位置付けられ得る。

【０１４０】
図４６に戻って、ＣＰＵ２００がユニット作業を送信してメインＣＰＵ２２８が受領した後、ユニットが存在する場所を決定するカーソル位置に基づいて検索が行われる（４４３）。ユニット作業は、特定のフレームに表示されたすべてのユニットと関連する。稼働中ユニットの中から、ユーザは、処理を行う対象ユニットを選択する。処理しようとするオブジェクトに関連するオブジェクト属性情報は、ＣＰＵ２２８に関連するオブジェクト属性メモリ内に書き込まれ、メモリは、修正されたユニットが表示されることになる場所を決定することができるように監視される。その後、対話ボックスが表示されて、上述したように修正されるユニットを識別して、重ね合わせ要求が行われて、それによってゲーム画面および対話ボックスの両方を表示することとなる（４４４）。その後、ユーザは、修正する特定の対象ユニットを選択する（４４５）。ユーザが対象ユニットを選択した後、さらに対話ボックスが現れて、ユーザはオブジェクトモードを選択する（４４６）。例えば、選択されるオブジェクトモードは「ドットアイコン」であってもよく、ドットアイコンでは、ユーザは、ドット毎にキャラクタ構成を変更することができる。

【０１４１】
ゲームＣＰＵ２００は、ループ内のデータ要求を待って（４３５）、最終的にはメインＣＰＵ２２８からの要求を受信する（４４７）。最新のユーザデータは、ステップ４３６に示すように、ＣＰＵ２００によって送信されて、メインＣＰＵ２２８はこのデータを取り入れて（４４８）、本明細書で説明しているエディタのうちの１つのを用いて、データを変更する（４４９）。

【０１４２】
ゲームＣＰＵ２００の側では、ゲームＣＰＵは、メインＣＰＵ２２８が編集している新規のデータユニットを待って（４３７）、ブロック（４３８）において、ゲームＣＰＵ２００に対してメインＣＰＵ２２８ルーチンのステップ（４５１）を介して送信される新規のデータユニットを受信する。ゲームＣＰＵ２００は、また、ユニットに対してなされた変更を識別する２バイトのデータブロックであるデータポインタを、ブロック４３８で受信

する。ブロック４３８で情報を受信した後、ゲームＣＰＵ２００は、再開処理へ移って、変更されて新規のユニット情報を識別するポインタによってゲームの実行を再開する。

【０１４３】

ブレイク命令から復帰すると、ゲームプログラムは、システムブレイクが生じたのと同じ点から継続する。ブレイクからの復帰は、所望する変更が行われた後に生じるＣＯＰ命令がもはや発せられなくなった後に開始する。メインＣＰＵ２２８は、分岐してブロック４４０に戻り、ループ内に留まって新たな停止要求を待つ。

【０１４４】

図４９および図５０は、ゲームプログラム処理が、本実施例に従った「ユニット作業」テーブル処理技術を用いてどのように生じるかを示すフローチャートである。メインルーチンゲームプログラム（４５０）が実行を開始すると、ゲームＣＰＵ２００が初期化されて（４５２）、初期値がユニット作業領域内に格納される。その後、デフォルトデータが、モデルソフトウェア内の元データに従って、ユニット作業領域にセットされる（４５４）。そのような初期データに基づいて、初期画面が生成される（４５６）。その後、メインテーブル処理サブルーチン４５８ならびにブランキングおよびビデオＲＡＭ復元処理（４６０）中に、すべてのメインルーチンが処理される。

【０１４５】

図５０は、メインテーブル処理（４５８）を示すフローチャートである。図４２のステージステータスに対応するメインテーブルは、ゲームシーケンスを示す。メインテーブルは、プログラム層の中で最上位のプログラムルーチンであり、図３８に示すモデルソフトウェアのフローは、メインテーブルによって実行される。図３８のプログラム処理４０９が実行されると、図４３のオブジェクトステータスに対応するユニット動作テーブルが、各ユニットに対して参照される。メインテーブル処理サブルーチン４５８では、まず、図５１で示すメインテーブルの例のようなメインテーブル内で識別された条件に対して参照が行われる（４６２）。図５１に示すように、メインテーブルは、「条件」列と「処理」列とを含む。条件列は、識別された条件に関連するサブルーチンをポイントする多くのルーチンアドレスを含む。図５０に戻って、ブロック４６４において、条件列内で識別された所定の条件と一致するかどうかを決定するチェックが行われる。条件チェックは、「ビンゴ（合致）」フラグの状態によって決定される。所定の条件と一致すれば、メインテーブルが、条件に対応して生じる処理を識別するために参照される。処理列内に示される処理操作は、処理列内で識別された処理に関連したルーチンアドレスポインタによってポイントされる処理を実行することによって遂行される。条件が一致しないことがブロック４６４でのチェックによって決定された場合は、メインテーブル処理ルーチンは分岐してブロック４６８に進み、メインテーブルが終了したどうかを決定するチェックが行われる。もし終了していなければ、ルーチンは分岐してメインテーブルの次の条件へ進み、メインテーブルのすべてのエントリが処理されるまで処理を継続して、すべてのエントリが処理されると、図４９に示すメインゲームプログラムルーチンへ戻る（４７０）。

【０１４６】

図５１のメインテーブルを再検討して、「必ず」条件に関連するエントリは、ユニットが一定の方法で動作しなければならない条件を定義する「必ず」ルーチンのアドレスをポイントする。図５２に示す必ずサブルーチン（４７５）に従って、「ビンゴ」と呼ばれるフラグが１にセットされ（４７７）、この条件は必ず生じて、ルーチンはメインテーブル処理ルーチンに戻ることとなる（４７９）。

【０１４７】

図５３は、メインテーブルに示す「敵全滅」条件サブルーチン（４８０）を示す。このルーチンは、敵が削除されるかどうかを決定して、「ビンゴ」フラグをセットする。図５３は、また、味方が全滅させられるかどうかを決定するための「味方全滅」条件サブルーチン（４８０）を示す。敵（または味方）全滅ルーチンに従って、最初に、敵（または味方）の数が０に等しいかどうかを決定するチェックが行われる（４８２）。敵（または味方）の数が０に等しい場合は、ビンゴフラグが１にセットされ（４８４）、ルーチンは分岐

してメインテーブル処理サブルーチンに戻る。次に、図５６の「次ステージへ」サブルーチン（または図５７の「エンディングへ」サブルーチン）が、図５０の処理４６６において実行される。敵（または味方）の数が０に等しくない場合には、ビンゴフラグが０にセットされる（４８６）。その後、ルーチンは、メインテーブル処理ルーチンへ戻る。対応する処理は実行されない。

【０１４８】

メインテーブル内の「ユニット動作」サブルーチン処理（５００）に従って、ユニット動作テーブルは、例えば、「亀」ユニット作業に関する動作を示す図５４に示すユニット動作テーブルの図と同様に処理される。ユニット動作テーブルで特定された条件に関して、各条件および各処理は、条件および処理に関わるサブルーチンをポイントするアドレスポインタによって識別される。「必ず」条件ルーチンは、図５２に関連して前述した。「必ず」に続く条件は、画面の右端に触れて亀が検出される時に適用する。「踏まれたら」条件は、亀が別のオブジェクトに踏まれる場合に適用する。左側の条件が一致すれば、関連するアドレスポインタによって示される処理が実行される。図５４に示すテーブル内の各条件に関連する処理は自明である。

【０１４９】

図５５に示す「ユニット動作」サブルーチン（５００）に従って、最初に、ルーチンが、図４５のユニット作業領域内のユニットデータを参照する（５０２）。その後、図５４に示すユニット動作テーブルの内容が、各条件に関連したルーチンアドレスによって示されるサブルーチンをアクセスすることによって処理される（５０４）。ユニット動作テーブルは、ユーザがステータスエディタを介して図２２に示す表示画面を用いることによって、修正されてもよい。その後、条件が一致したかどうかを決定するためのチェックが行われる（５０６）。ブロック５０６のチェックにより、条件が一致したことが示されると、図５４のテーブルの処理部が、例えば亀を右に移動させるような関連処理を遂行するためにアクセスされる（５０８）。ユニット動作テーブルに関連する第１の条件が一致しない場合は、ルーチンは分岐してブロック５１０に進み、エンドマークを検出することによって、テーブルが終了状態かどうかを決定するチェックが行われる。テーブルが終了していない場合は、ルーチンは分岐してブロック５０４に戻り、処理が継続する。テーブルが終了状態にある場合は、５１２において、処理するユニットがこれ以上存在しないかどうかを決定するチェックが行われる。このチェックにより、処理するユニットがまだ存在することが示されると、ルーチンは分岐してブロック５０２に戻る。処理するユニットがこれ以上存在しない場合は、ルーチンは呼出ルーチンに戻る（５１４）。

【０１５０】

（図５１の）メインテーブルに戻って、「敵全滅」条件が検出されると、図５６に示す「次のステージへ」ルーチンが実行される（５１６）。最初に、ステージ番号が増分されて、次のステージ用に初期化する（５１８）。その後、図４５のユニット作業領域がクリアされ（５２０）、いくつかのユニットがセットされて、背景ピクチャが次のステージ用に表示される（５２２）。次のメインルーチンは、次ステージルーチンにセットされる。その後、ルーチンは分岐して呼び出しルーチンに戻る（５２６）。メインテーブルに示すの最終処理は「エンディングへ」処理で、この処理は、味方全滅条件の検出によって実行される。図５７に示すように、「エンディングへ」サブルーチンが呼び出されると（５２８）、ユニット作業がクリアされる（５３０）。その後、ユニットはエンディング用に向けてセットされ（５３２）、関連ピクチャが表示される（５３４）。その後、ルーチンは分岐して呼び出しルーチンに戻る（５３６）。次のメインルーチンは、エンディングステージルーチンへセットされる。

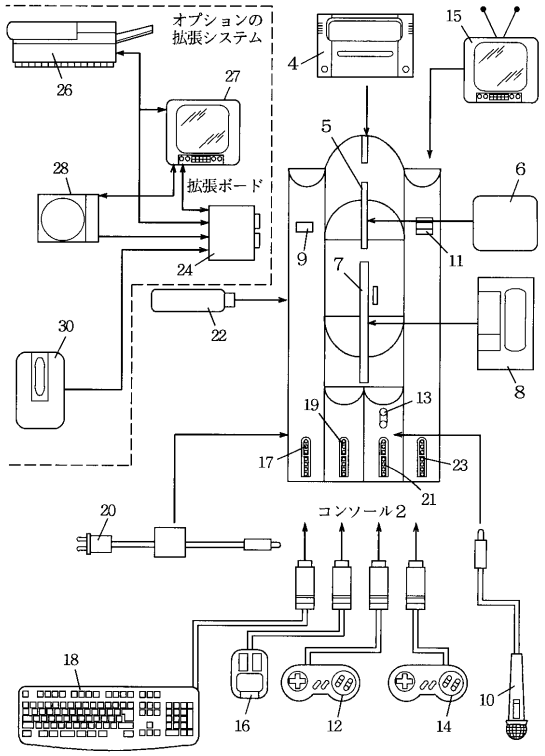
【０１５１】

本発明は、現時点において最も実用的かつ好ましい実施例と思われるものに関して説明してきたが、本発明は開示された実施例に限定されるものではなく、反対に、付加されたクレームの精神および範囲内に含まれる種々の修正および構成を含むことが意図されていることが理解されるべきである。

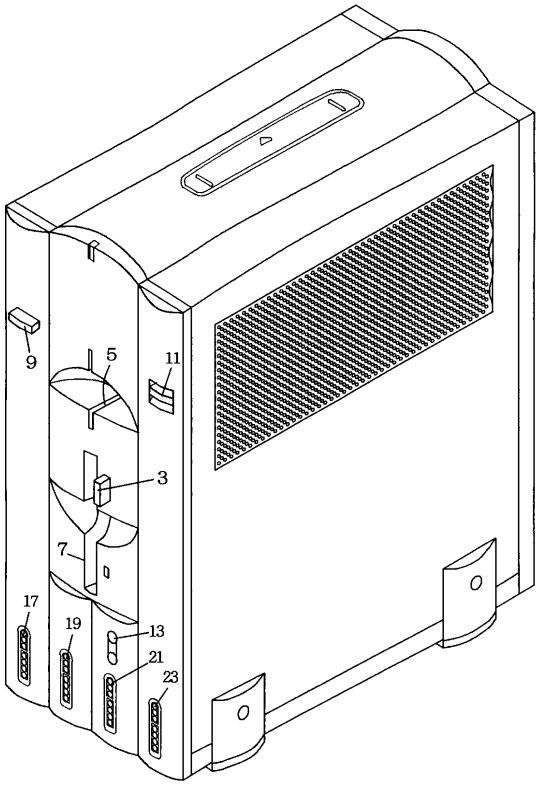
【図面の簡単な説明】
【図１】ビデオグラフィックゲーム作成システムのブロック図である。
【図２】図１に示すゲームプロセッサシステムコンソールユニット２の斜視図である。
【図３】図１に示すゲームプロセッサシステムの実施例のより詳細なブロック図である。
【図４】図１に示すゲームプロセッサシステムの実施例のより詳細なブロック図である。
【図５】システムメモリスペースの一部のメモリマップを示す図である。
【図６】システムメモリスペースの一部のメモリマップを示す図である。
【図７】図３および図４の実施例が用いられ得るオンラインネットワーキングシステムの例を示す図である。
【図８】本実施例のプログラム作成ツールをアクセスするためのタイトル画面の例を示す図である。
【図９】システムブレイク画面の例を示す図である。
【図１０】キャラクタ編集画面の例を示す図である。
【図１１】ユーザがキャラクタアイコンをクリックするときアクセスされる画面の例を示す図である。
【図１２】「マリオ」という名称のユニットのための「全」ステータス画面表示の例を示す図である。
【図１３】キー編集画面の例を示す図である。
【図１４】ユニット動画編集画面の例を示す図である。
【図１５】背景画面エディタの例を示す図である。
【図１６】「マップエディタ」の例を示す図である。
【図１７】ゲーム表示画面の背景が、背景画面をドラッグして動作中の背景上に貼り付けることによって、どのように修正されるかを示す図である。
【図１８】音楽編集画面を示す図である。
【図１９】音声コース毎に音楽属性を変更することができる音楽エディタ画面の例を示す図である。
【図２０】音楽を再生中にリアルタイムで変更することができる音楽エディタ画面の例を示す図である。
【図２１】オートプログラマの操作を示す画面表示の例を示す図である。
【図２２】ステータスエディタ表示画面の例を示す図である。
【図２３】幅広い範囲の背景キャラクタ編集が可能な「ステージ」ウインドウ表示画面の例を示す図である。
【図２４】図８に示すマリオファクトリーのタイトル画面の機能を示す図である。
【図２５】「ツールボックス」選択画面を介してアクセス可能な編集関連ツールを識別する機能図である。
【図２６】ゲームブレイク中にアクセス可能なモデルソフトウェア関連編集機能を識別する機能図である。
【図２７】図３に示すゲートアレイ２２６の詳細なブロック図である。
【図２８】重ね合わせ操作器回路を示す図である。
【図２９】ＲＡＭカートリッジの実施例の簡単な概要図である。
【図３０】カートリッジのアースコネクタ、アドレスコネクタ、データコネクタ、リセットコネクタおよび電源コネクタを示す図である。
【図３１】ＲＡＭカートリッジの実施例の簡単な概要図である。
【図３２】カートリッジのアースコネクタ、アドレスコネクタ、データコネクタ、リセットコネクタおよび電源コネクタを示す図である。
【図３３】本発明の実施例に従ったリセット関連回路のブロック図である。
【図３４】本発明の実施例に従ったシステムブレイクハードウェアのブロック図である。
【図３５】電源の立ち上げから「マリオファクトリー」ユーティリティータイトル画面が現れるまでの、本発明の実施例に従った処理のシーケンスを示すフローチャートである。
【図３６】ユニット作業関連表示画面の例を示す図である。

【図３７】「ユニット」を基本としてプログラミングがどのように構築されているかを機能的に示すブロック図である。
【図３８】ユニットデータの処理と、ピクチャ信号および音声信号のユーザの表示画面への出力とに含まれる処理のシーケンスを示すフローチャートである。
【図３９】ユニットデータの処理と、ピクチャ信号および音声信号のユーザの表示画面への出力とに含まれる処理のシーケンスを示すフローチャートである。
【図４０】ユニットデータの処理と、ピクチャ信号および音声信号のユーザの表示画面への出力とに含まれる処理のシーケンスを示すフローチャートである。
【図４１】ユニットデータの処理と、ピクチャ信号および音声信号のユーザの表示画面への出力とに含まれる処理のシーケンスを示すフローチャートである。
【図４２】第１および第２のモデルソフトウェアフォーマットの例を示す図である。
【図４３】第１および第２のモデルソフトウェアフォーマットの例を示す図である。
【図４４】オブジェクトユニットＲＡＭ情報領域とゲーム背景関連ＲＡＭ領域とを含むＲＡＭ情報内容のメモリマップの例を示す図である。
【図４５】メインＣＰＵ２２８およびゲームＣＰＵ２００のメモリマップの例を示す図である。
【図４６】ゲーム作成処理中にゲームＣＰＵおよびメインＣＰＵ２２８が遂行する処理のシーケンスを示すフローチャートである。
【図４７】稼働中のユニットのためのユニットポインタ領域と、識別された稼働中のユニットのための代表的なユニット情報とを示す図である。
【図４８】ユニットデータの例を識別するすべてのユニットのためのユニットポインタ領域を示す図である。
【図４９】ゲームプログラム処理が「ユニット作業」テーブル処理技術を用いてどのように生じるかを示すフローチャートである。
【図５０】ゲームプログラム処理が「ユニット作業」テーブル処理技術を用いてどのように生じるかを示すフローチャートである。
【図５１】メインテーブルの例を示す図である。
【図５２】「必ず」条件処理サブルーチンの例を示す図である。
【図５３】「敵全滅」条件サブルーチンの例を示す図である。
【図５４】「ユニット動作」テーブルの例を示す図である。
【図５５】「ユニット動作」サブルーチンの例を示す図である。
【図５６】「次のステージへ」ルーチンの例を示す図である。
【図５７】「エンディングへ」サブルーチンの例を示す図である。
【符号の説明】
２…ゲームプロセッサシステムユニットコンソール
４…ＲＡＭカセット
５…挿入ポート
６…ゲームプロセッサ識別カード
８…フロッピーディスク
１０…マイクロフォン
１２，１４…２人プレイヤ操作器
１６…マウス
１８…キーボード
２０…交流アダプタ
２２…モデム
２４…拡張ボード
２６…プリンタ
２８…ハードディスクドライブ
３０…スキャナ

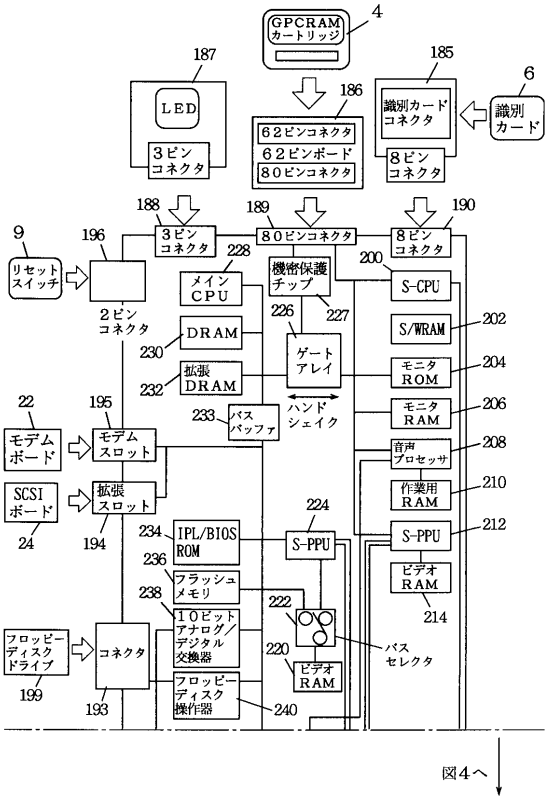
【図１】


オプションの
拡張システム
拡張ボード
コンソール２

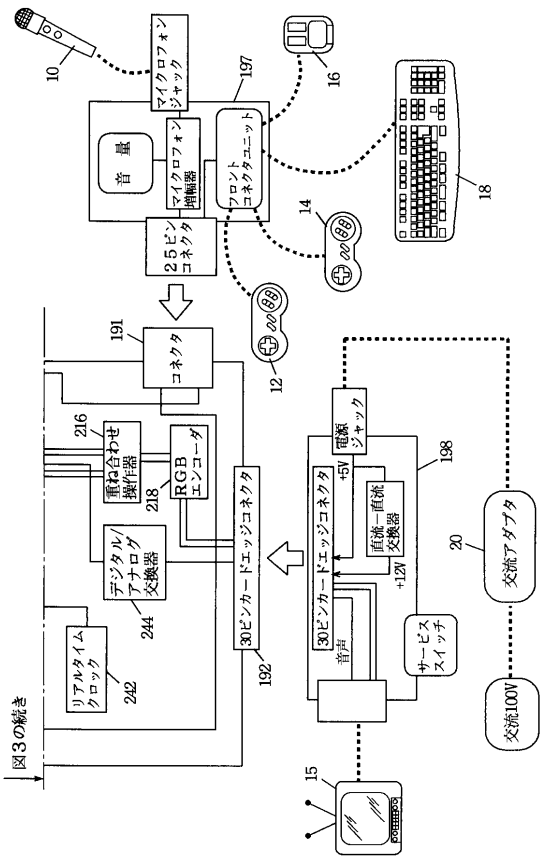
【図２】



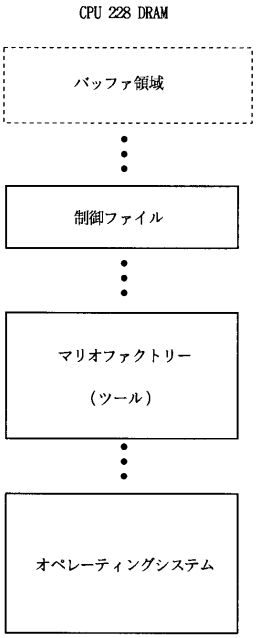
【図３】


GPCRAM
カートリッジ
LED
３ピン
コネクタ
６２ピンコネクタ
６２ピンボード
８０ピンコネクタ
識別カード
コネクタ
識別
カード
８ピン
コネクタ
リセット
スイッチ
２ピン
コネクタ
メイン
CPU
機密保護
チップ
S-CPU
DRAM
ゲート
アレイ
S/WRAM
拡張
DRAM
モニタ
ROM
バス
バッファ
ハンド
シェイク
モニタ
RAM
モデム
ボード
モデム
スロット
音声
プロセッサ
SCSI
ボード
拡張
スロット
作業用
RAM
IPL/BIOS
ROM
S-PPU
フラッシュ
メモリ
ビデオ
RAM
１０ビット
アナログ/
デジタル
交換器
バス
セレクタ
フロッピー
ディスク
ドライブ
コネクタ
ビデオ
RAM
フロッピー
ディスク
操作器
図４へ

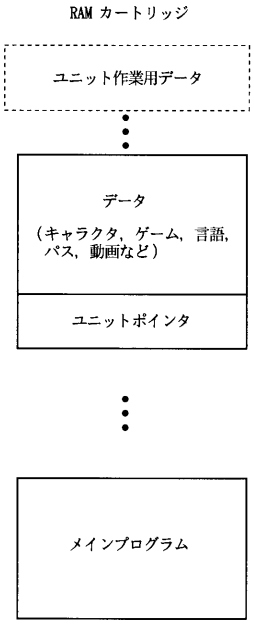
【図４】


マイクロフォン
ジャック
音量
マイクロフォン
増幅器
フロント
コネクタユニット
２５ピン
コネクタ
コネクタ
重ね合わせ
操作器
RGB
エンコーダ
デジタル/
アナログ
交換器
リアルタイム
クロック
３０ピンカードエッジコネクタ
電源
ジャック
+5V
直流－直流
交換器
+12V
音声
サービス
スイッチ
交流アダプタ
交流100V
図３の続き

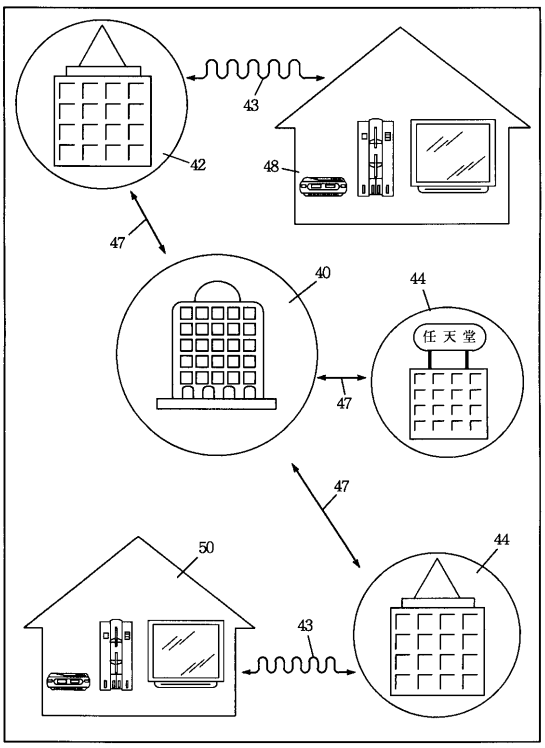
【図５】


CPU 228 DRAM
バッファ領域
制御ファイル
マリオファクトリー
（ツール）
オペレーティングシステム

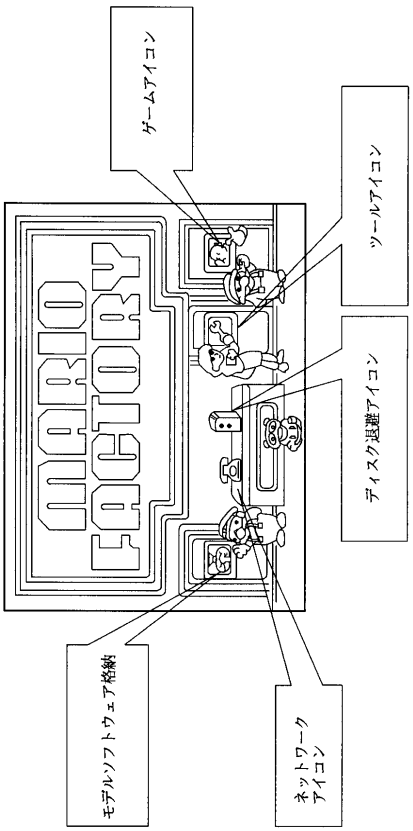
【図６】


RAM カートリッジ
ユニット作業用データ
データ
（キャラクタ，ゲーム，言語，
パス，動画など）
ユニットポインタ
メインプログラム

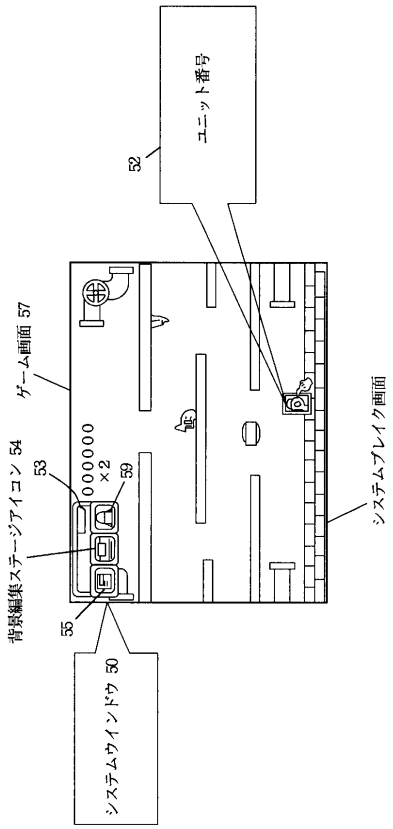
【図７】


43
42
48
47
40
44
任天堂
47
47
50
44
43

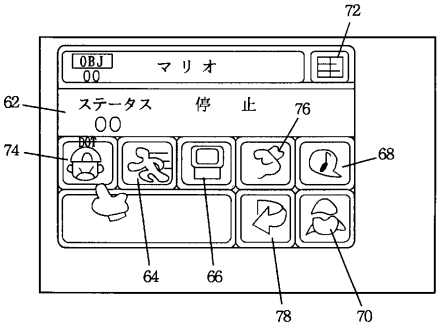
【図８】


ゲームアイコン
ツールアイコン
ディスク退避アイコン
モデルソフトウェア格納
ネットワーク
アイコン

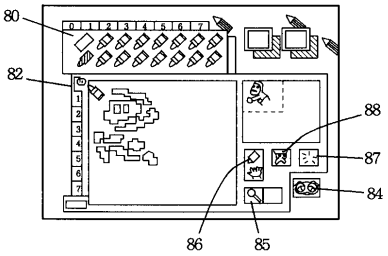
【図９】



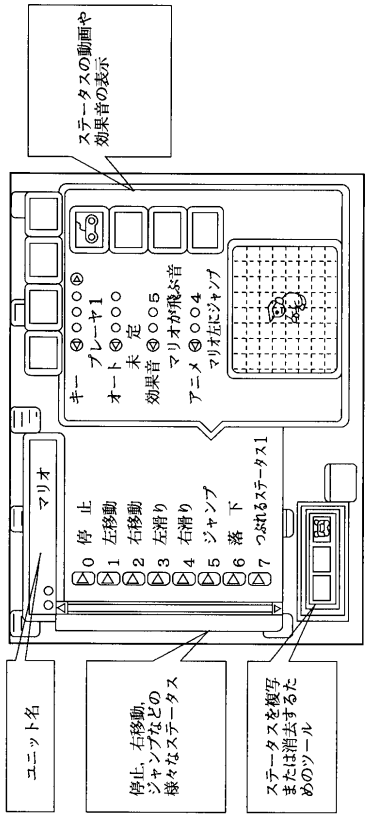
【図１０】



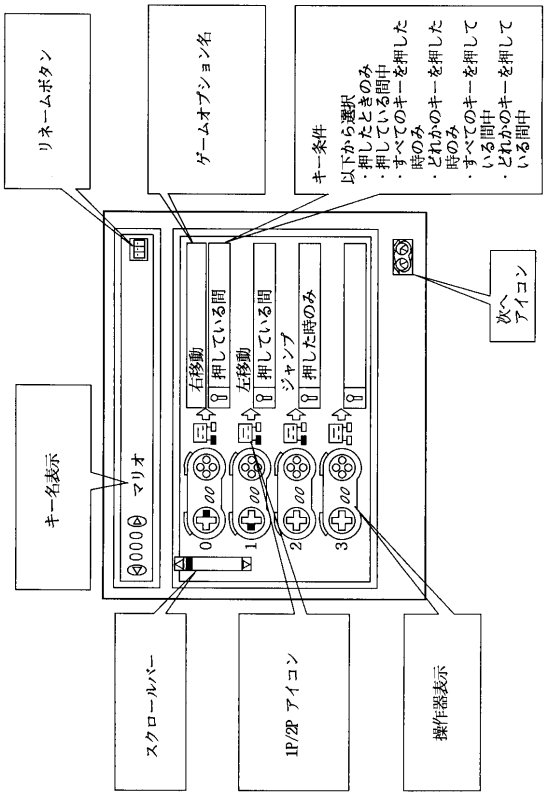
【図１１】



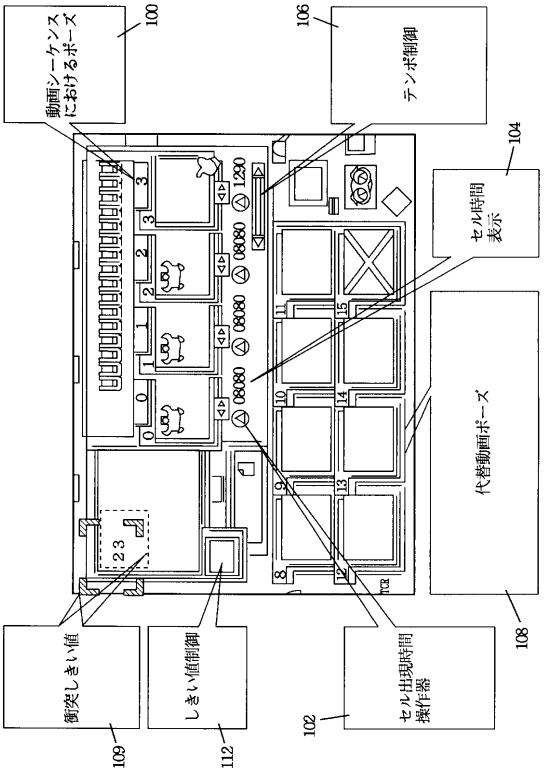
【図１２】



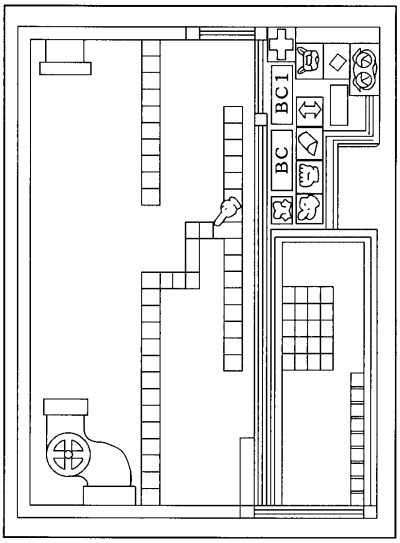
【図１３】



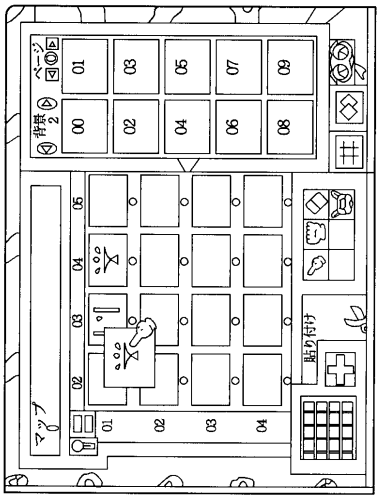
【図１４】



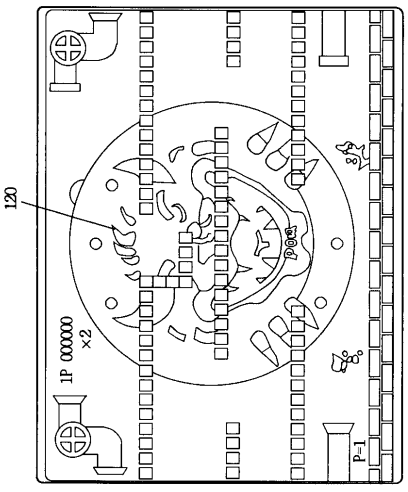
【図１５】



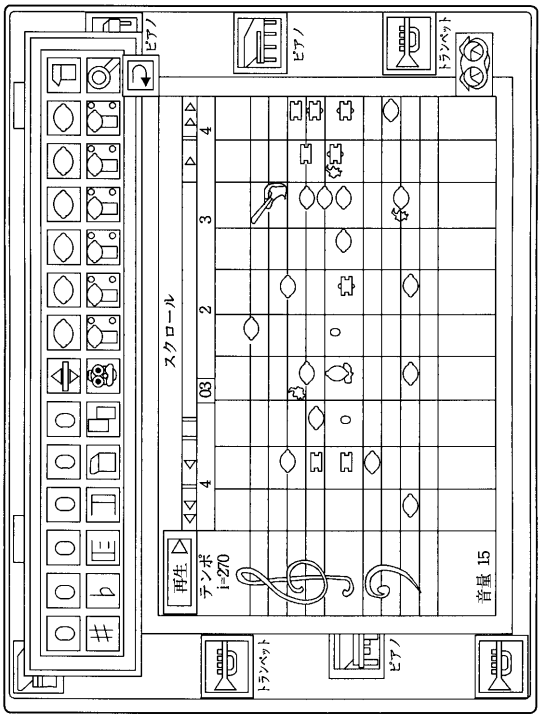
【図１６】



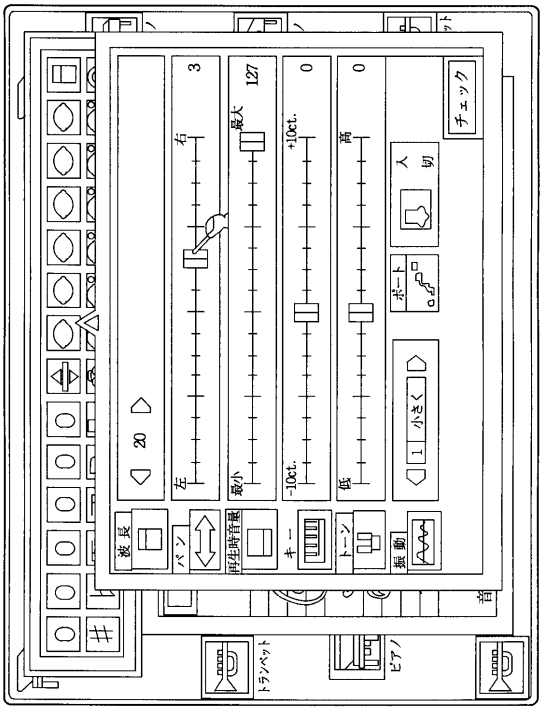
【図１７】



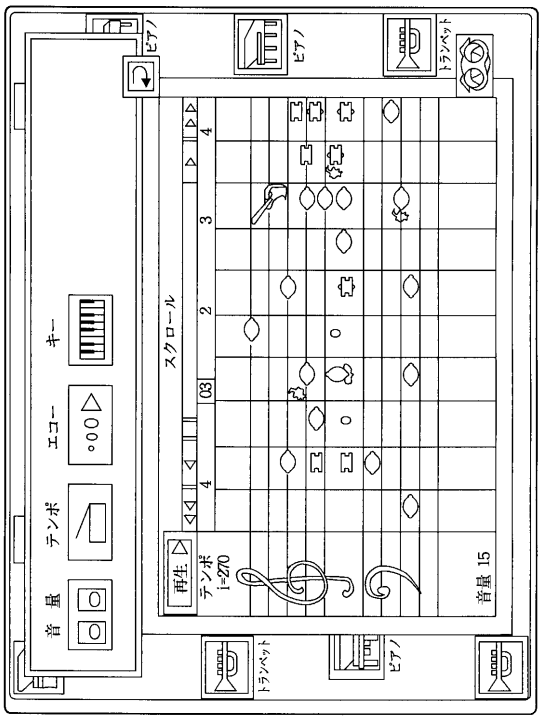
【図１８】



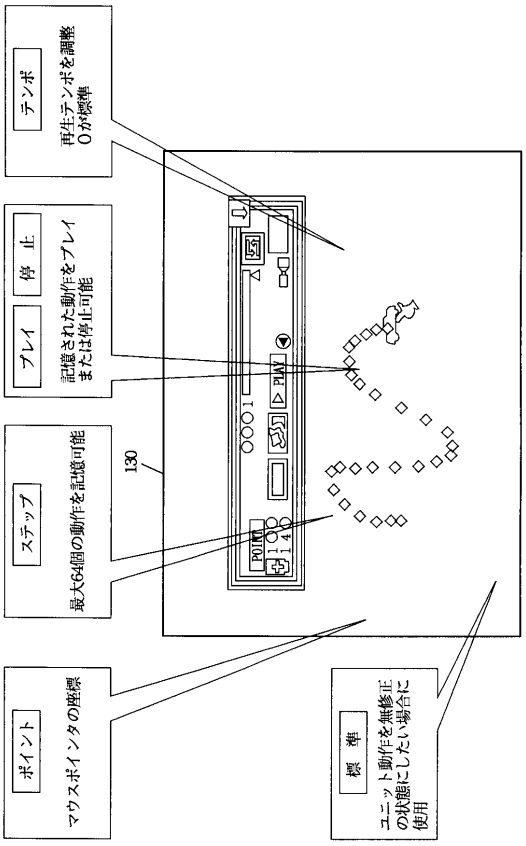
【図１９】



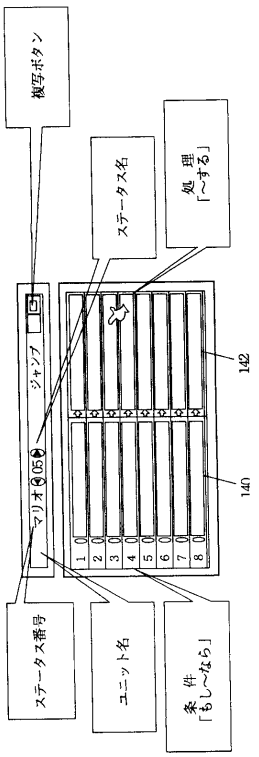
【図２０】



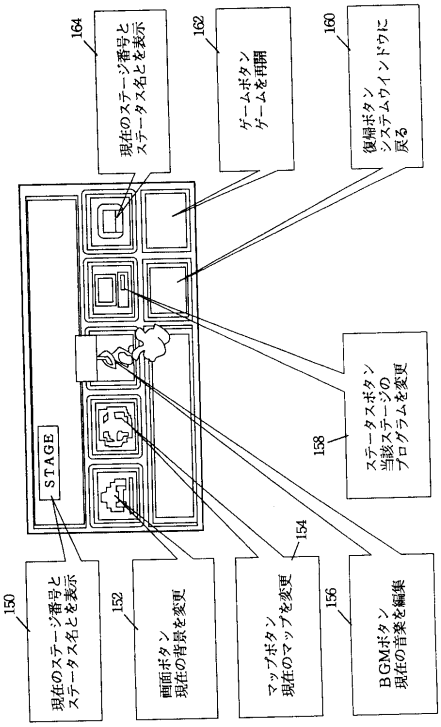
【図２１】



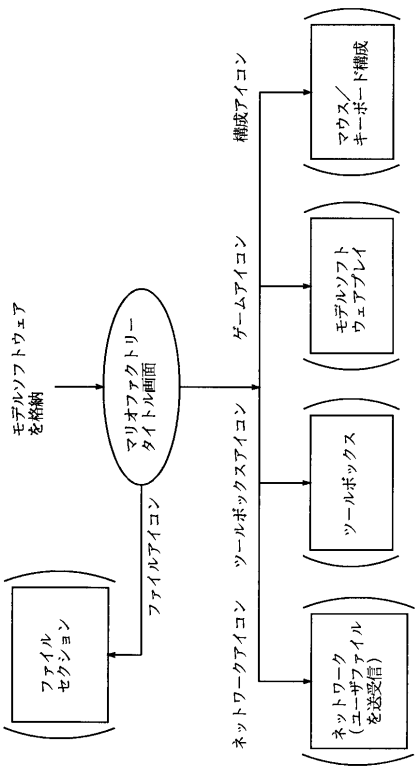
【図２２】



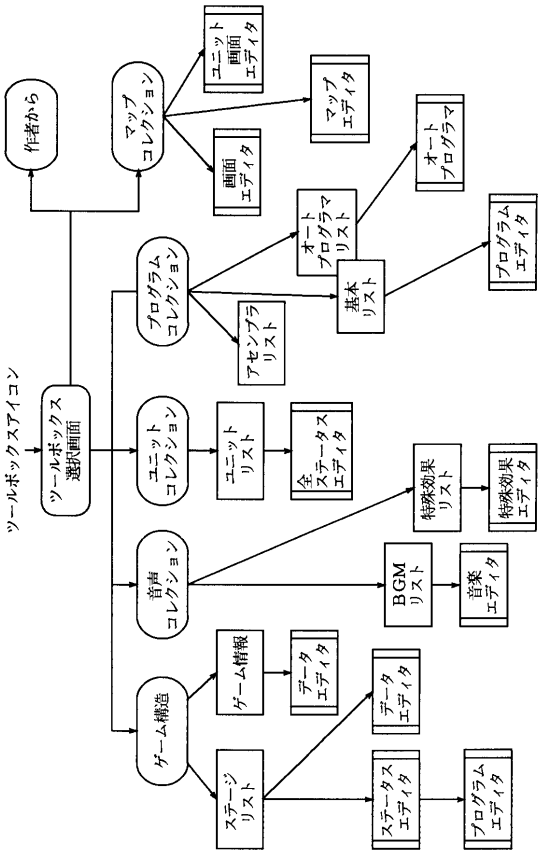
【図２３】



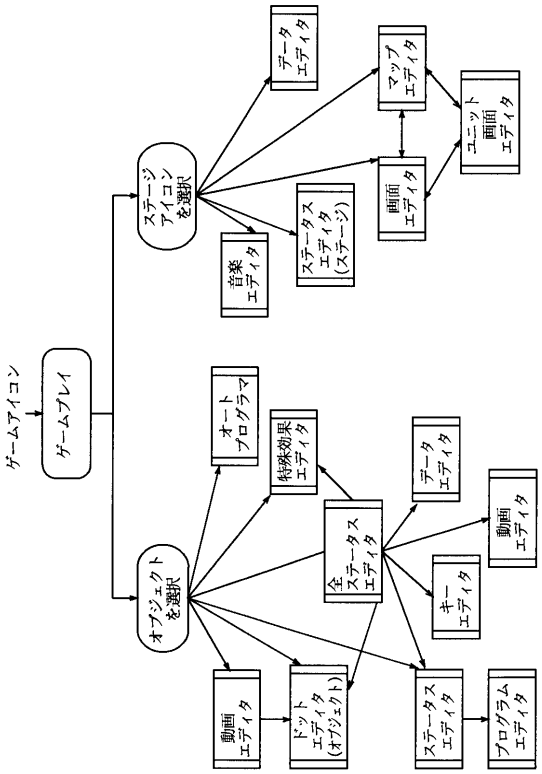
【図２４】



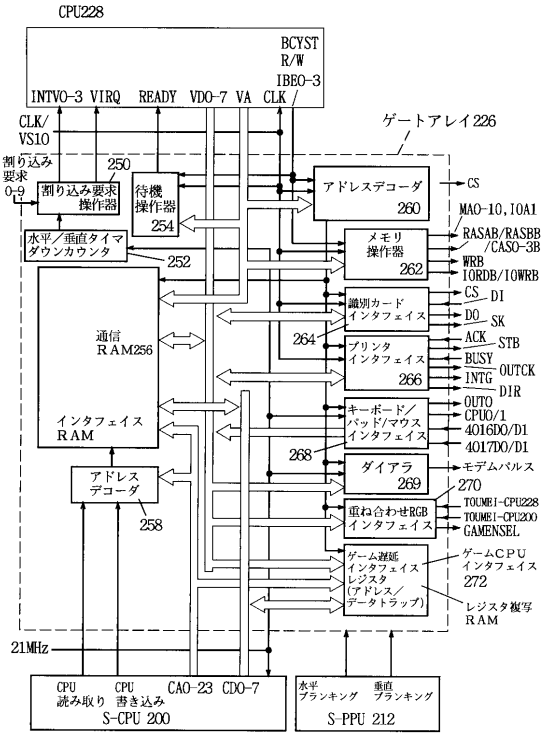
【図２５】



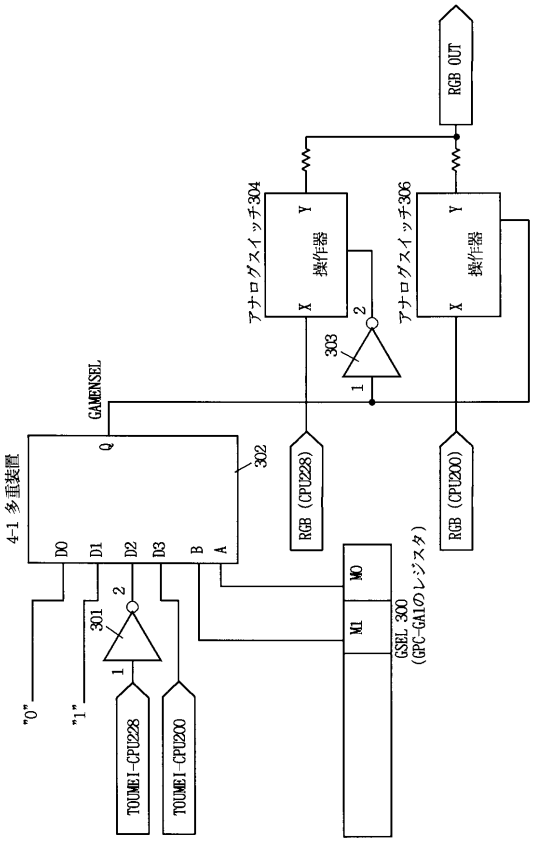
【図２６】



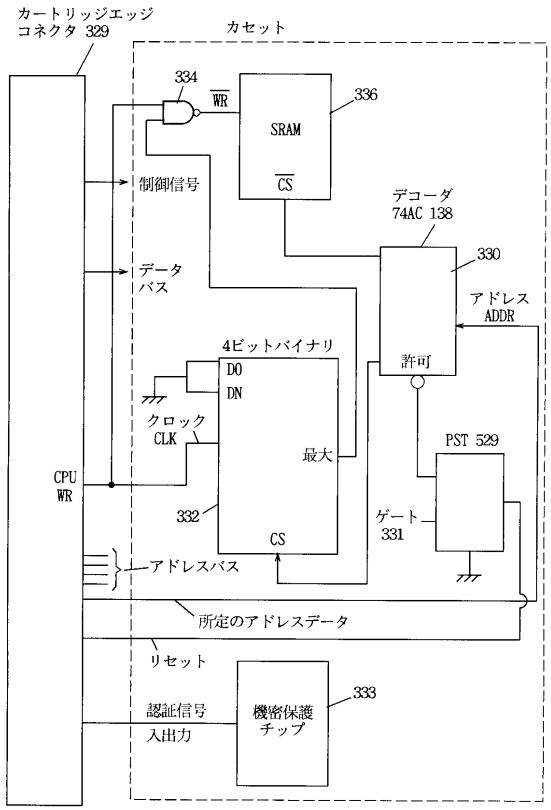
【図２７】



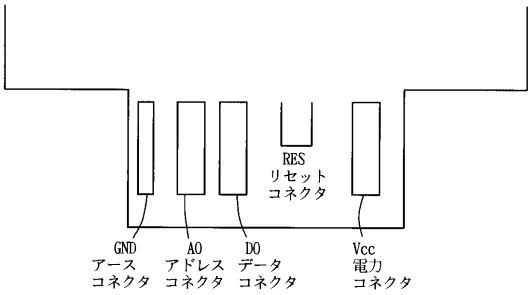
【図２８】



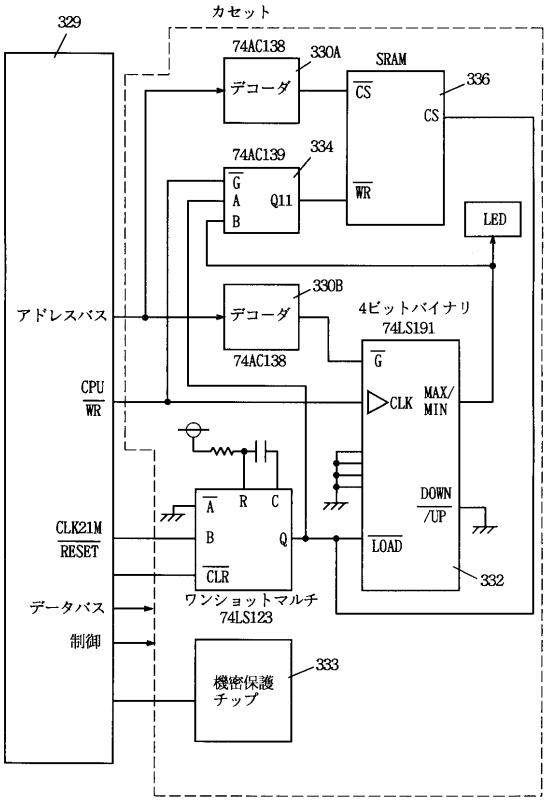
【図２９】



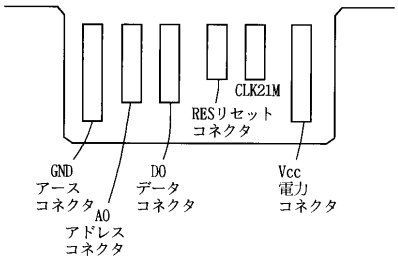
【図３０】



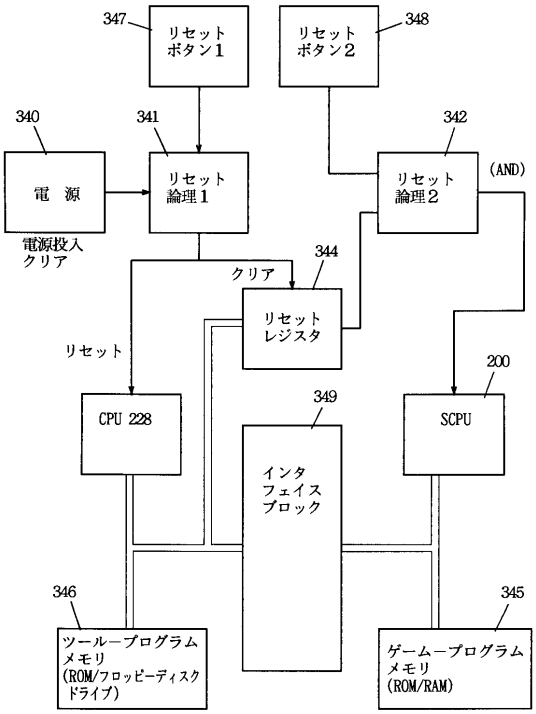
【図３１】



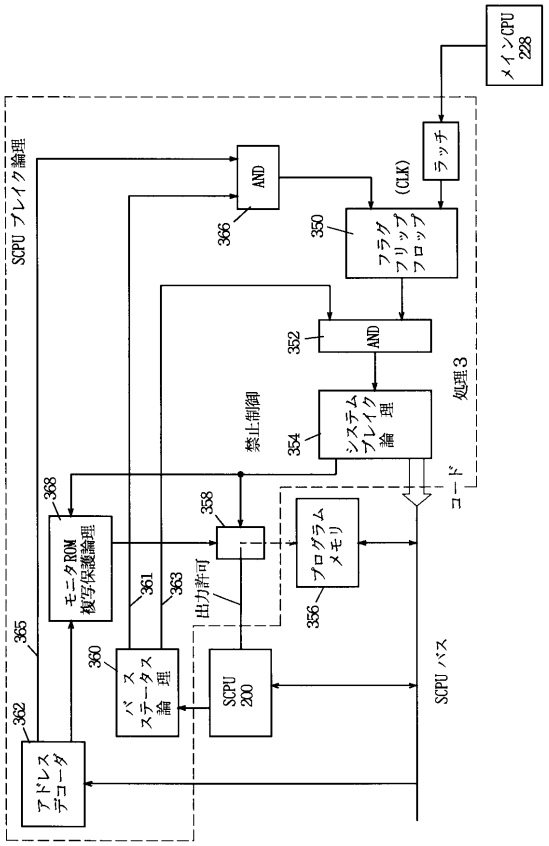
【図３２】



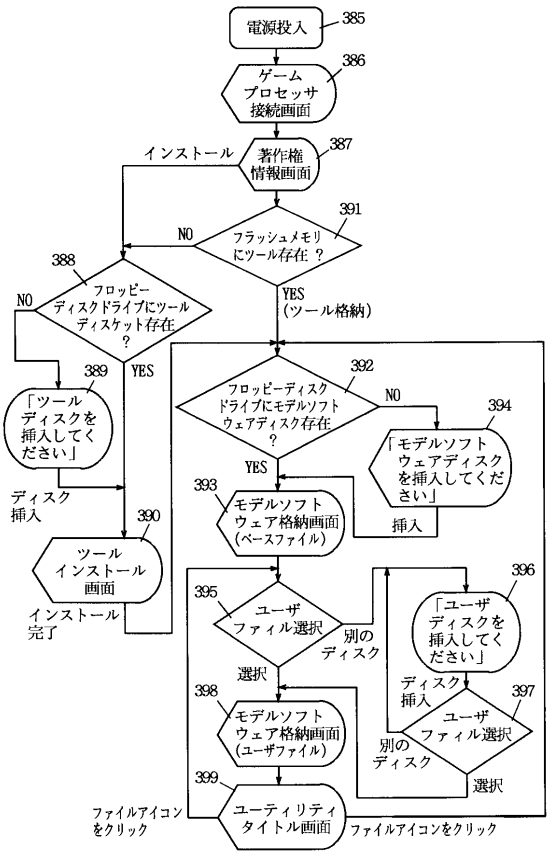
【図３３】



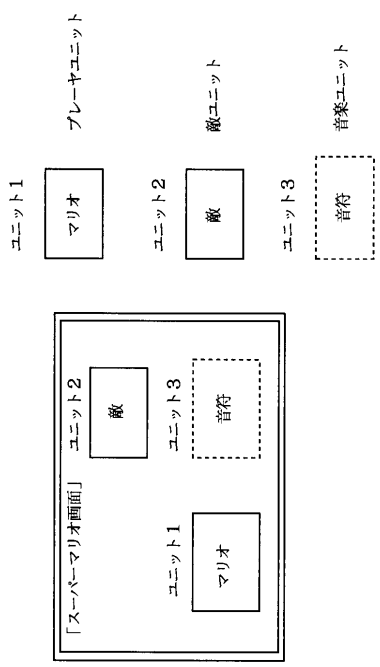
【図３４】



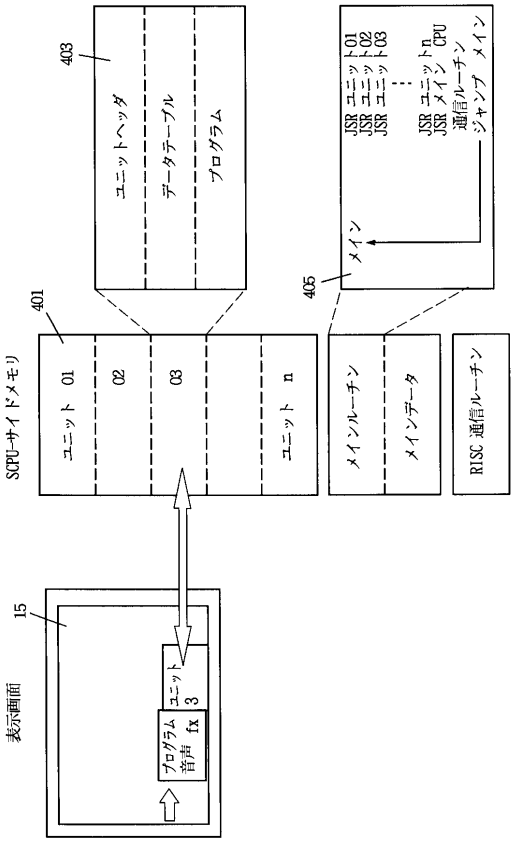
【図３５】



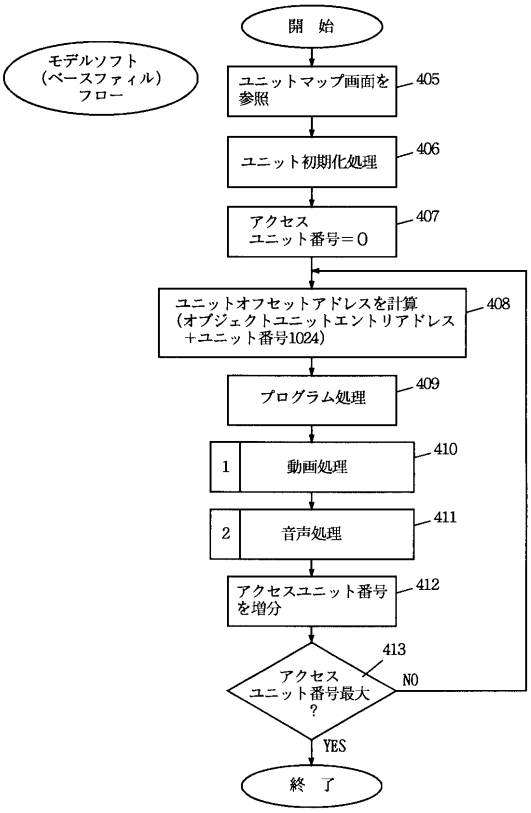
【図３６】



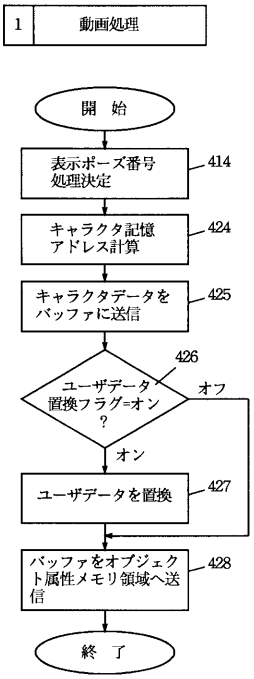
【図３７】



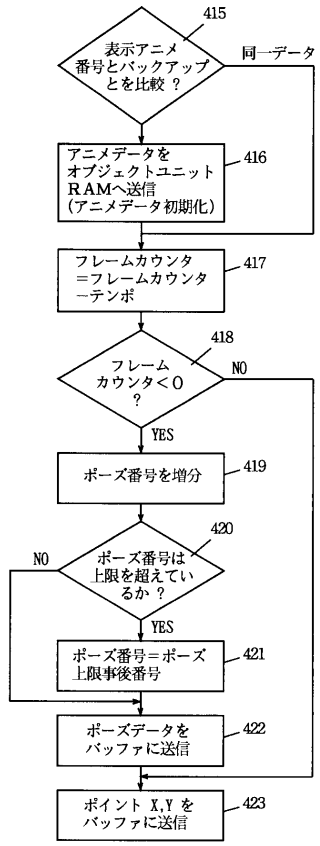
【図３８】



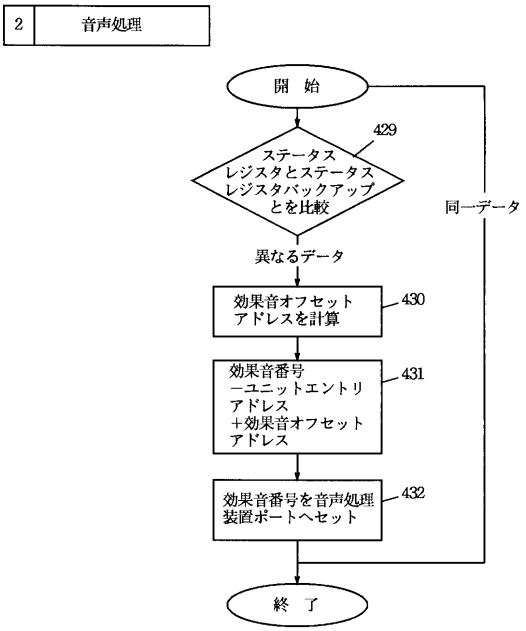
【図３９】



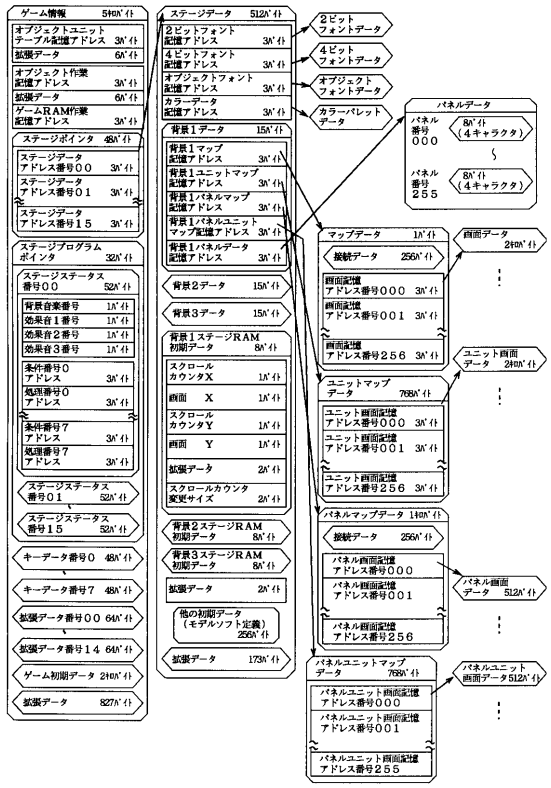
【図４０】



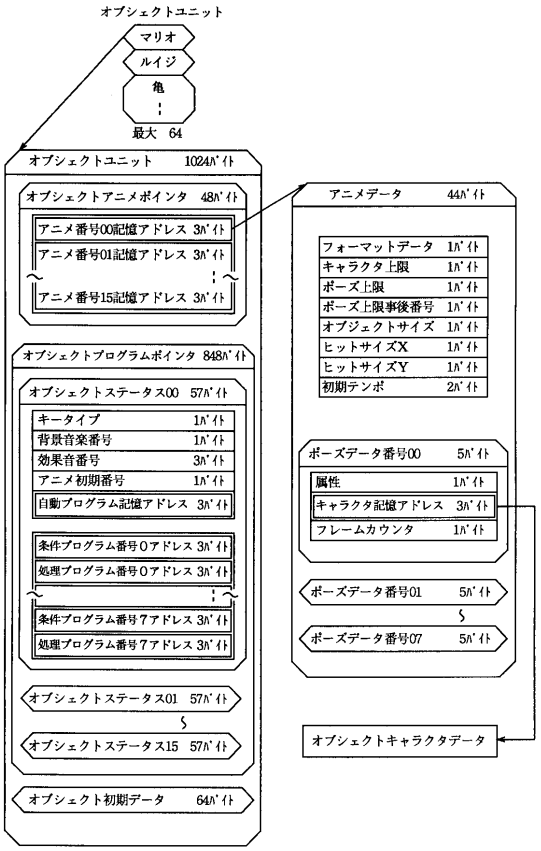
【図４１】



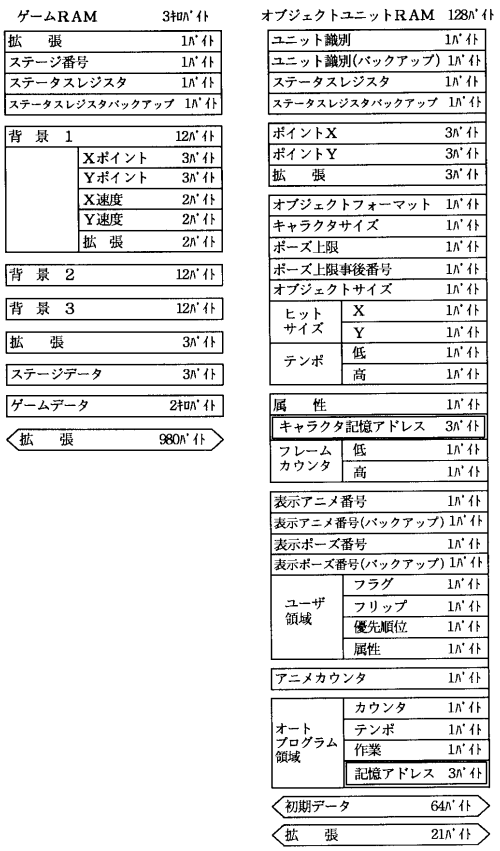
【図４２】



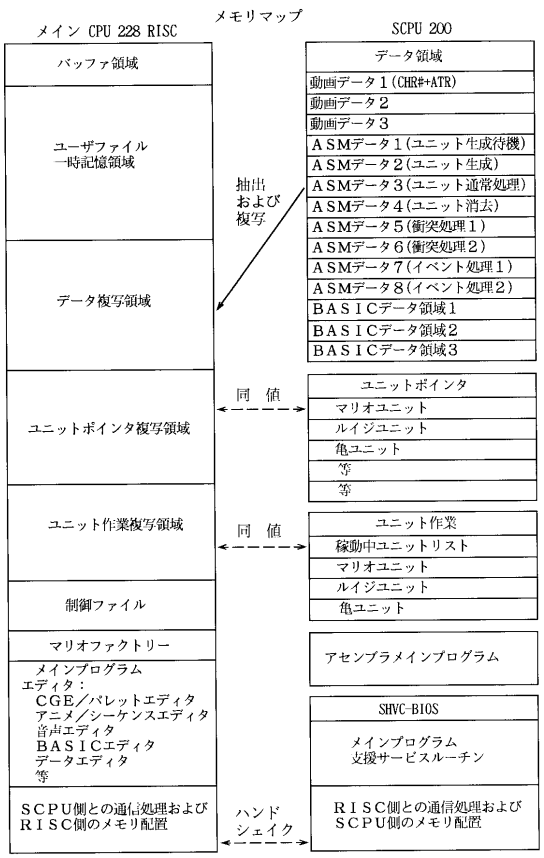
【図４３】



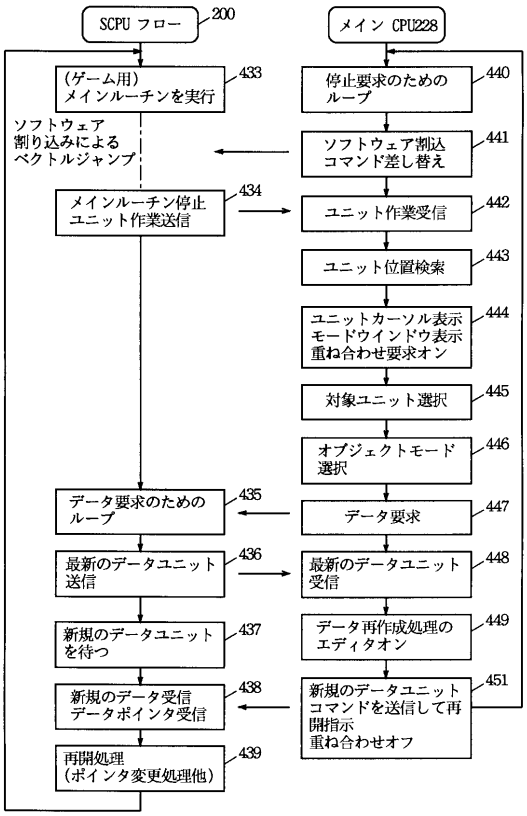
【図４４】



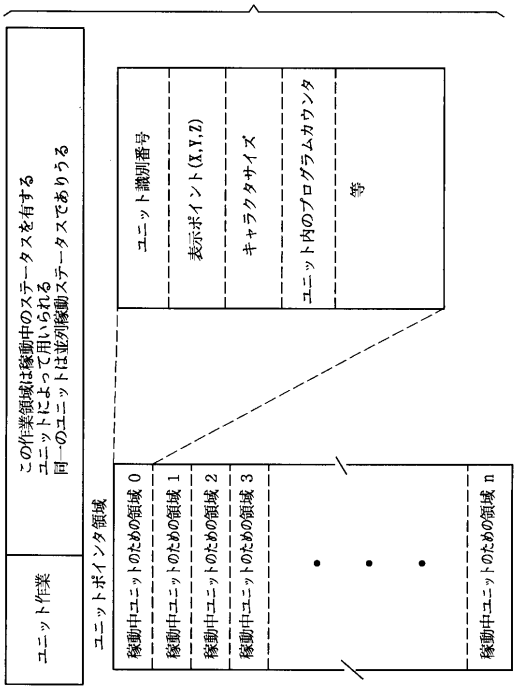
【図４５】



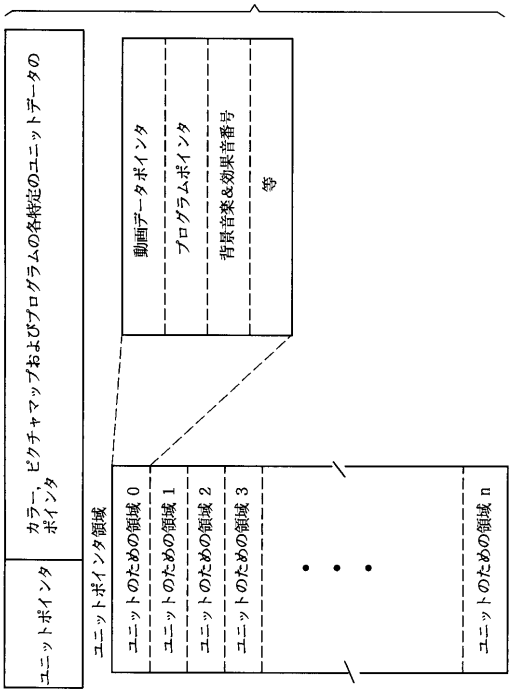
【図４６】



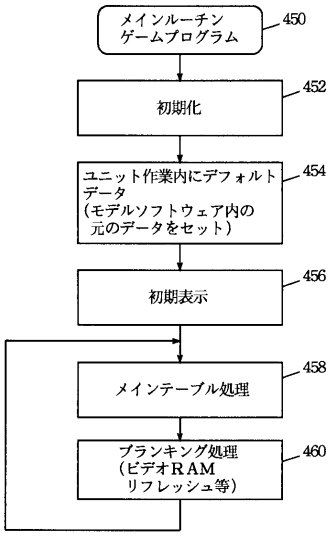
【図４７】



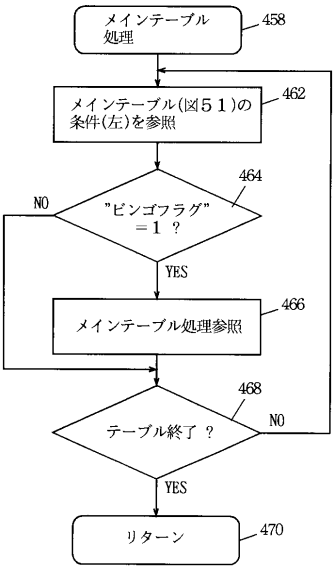
【図４８】



【図４９】



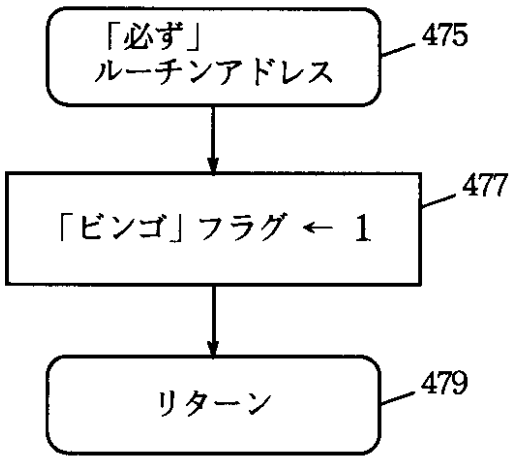
【図５０】



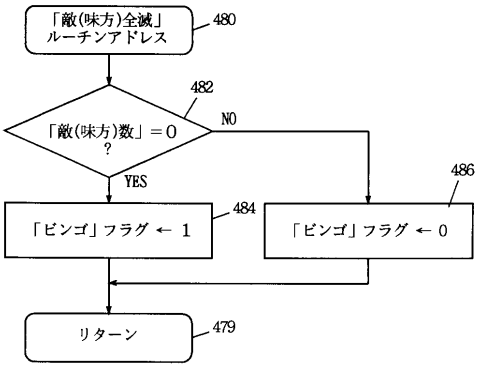
【図５１】

メインテーブル	処　理	条　件
	「ユニット動作」ルーチンアドレス	「必ず」ルーチンアドレス
	「次ステージへ」ルーチンアドレス	「敵全滅」ルーチンアドレス
	「エンディングへ」ルーチンアドレス	「味方全滅」ルーチンアドレス
	処理なし	エンドマーク

【図５２】



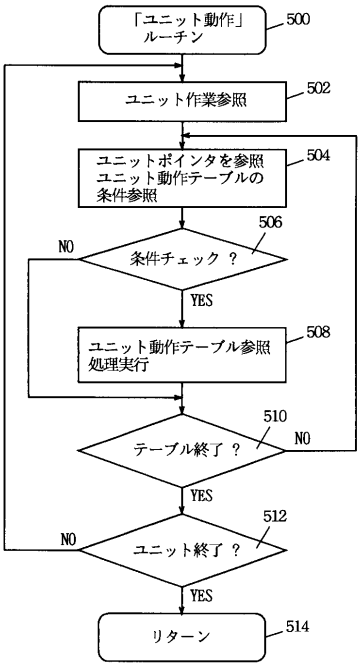
【図５３】



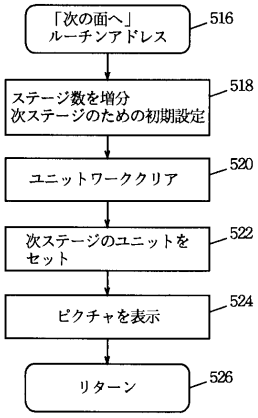
【図５４】

ユニット動作テーブル(亀の場合)	条件	処理
	「必ず」ルーチンアドレス	「右移動」ルーチンアドレス
	「右についたら」ルーチンアドレス	「左へワープ」ルーチンアドレス
	「踏まれたら」ルーチンアドレス	「踏まれたら」ルーチンアドレス
	エンドマーク	処理なし

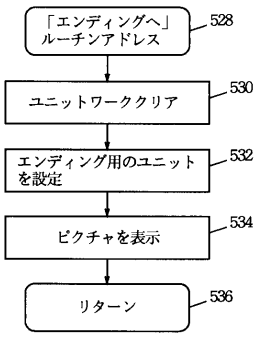
【図５５】



【図５６】



【図５７】



フロントページの続き

(51)Int.Cl.　　FI
　　G09G　5/36　　520N

(72)発明者　堀田　拓司
　　京都府東山区福稲上高松町６０番地　任天堂株式会社内
(72)発明者　藤原　和宏
　　京都府東山区福稲上高松町６０番地　任天堂株式会社内

審査官　橋爪　正樹

(56)参考文献　特開平０６－１１０４４８（ＪＰ，Ａ）
　　特開平０１－２３６３７７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
　　G06T11/00-17/50
　　A63F13/00-13/10