

【公報種別】特許法第 17 条の 2 の規定による補正の掲載

【部門区分】第 6 部門第 3 区分

【発行日】平成20年4月3日(2008.4.3)

【公開番号】特開2001-266050(P2001-266050A)

【公開日】平成13年9月28日(2001.9.28)

【出願番号】特願2001-42009(P2001-42009)

【国際特許分類】

G 0 6 F 19/00 (2006.01)

G 0 6 F 17/50 (2006.01)

【 F I 】

G 0 6 F 19/00 1 1 0

G 0 6 F 17/50 6 1 2 A

【手続補正書】

【提出日】平成20年2月19日(2008.2.19)

【手続補正 1】

【補正対象書類名】明細書

【補正対象項目名】特許請求の範囲

【補正方法】変更

【補正の内容】

【特許請求の範囲】

【請求項 1】 シミュレーションを実行するためにコンピュータシステムで使用されるコンピュータプログラムプロダクトであって、

コンピュータで読取り可能な複数のサービスプログラムコード手段であって、前記コンピュータで読取り可能なサービスプログラムコード手段は、シミュレーションされる処理の下で環境のオブジェクトのシミュレーションされる属性を集合的に決定するように構成され、各サービスプログラムコード手段は、オブジェクトコンテキスト内のオブジェクト属性の少なくとも 1 つのサブセットに関連づけられ、前記コンピュータで読取り可能なサービスプログラムコード手段の中の少なくとも幾つかは、前記サービスプログラムコード手段による前記オブジェクトコンテキストが認識可能な前記属性の識別名に基づいて行われる前記サービスプログラムコード手段と前記オブジェクトコンテキストとの間の相互通信を行うため、および、前記オブジェクト属性に対して処理を行うため、前記サービスプログラムコード手段を前記オブジェクトコンテキスト中の前記属性に結合する、コンピュータで読取り可能な属性アクセスプログラムコード手段を備える、コンピュータで読取り可能な複数のサービスプログラムコード手段と、

前記サービスプログラムコード手段のうちの少なくとも幾つかを前記オブジェクトコンテキストに結合することによって、前記オブジェクトコンテキストが認識できないユーザによって表現された属性名を前記オブジェクトコンテキストが認識可能な前記属性の識別名へマッピングするコンピュータで読取り可能なマッププログラムコード手段と、を備えることを特徴とするプログラムプロダクト。

【請求項 2】 前記マッププログラムコード手段は、前記ユーザによって表現された属性名である文字列を前記オブジェクトコンテキストが認識可能な属性の識別名に関連づけるためのプログラムコード手段を有する請求項 1 に記載のプログラムプロダクト。

【請求項 3】 ユーザによって識別される属性名を含む数学的式の文字列表現を前記オブジェクトコンテキストが認識可能な属性の識別名を含む数学的式の表現へ変換するためのコンピュータで読取り可能な式構文解析プログラムコード手段をさらに備える請求項 1 に記載のプログラムプロダクト。

【請求項 4】 前記式構文解析プログラムコード手段が変換する表現は、数学的機能が構成要素であるツリー構造を有する請求項 3 に記載のプログラムプロダクト。

【請求項 5】 ユーザによって識別される属性名から前記オブジェクトコンテキストの一部としての属性コンテキストへのマップを作成するためのプログラムコード手段と、前記属性コンテキストに前記属性に対する値を格納するためのプログラムコード手段と、を含む、属性を作成するためのコンピュータで読取り可能なプログラムコード手段を更に備え、

前記マッププログラムコード手段は、前記サービスプログラムコード手段のうちの少なくとも幾つかを前記属性コンテキストに結合することと、前記ユーザによって識別される属性名を前記属性コンテキスト中の対応する前記属性の値へマッピングすることと、をさらに行う、請求項 1 に記載のプログラムプロダクト。

【請求項 6】 各サービスプログラムコード手段は、別のサービスプログラムコード手段とは独立した速度で実行され、

前記プログラムプロダクトは、各サービスプログラムコード手段が前記オブジェクトコンテキスト中の前記オブジェクト属性全部の首尾一貫した表示をもつように前記サービスプログラムコード手段と前記オブジェクトコンテキストとの間の前記相互通信を調整するためのコンピュータで読取り可能なプログラムコード手段をさらに備える、請求項 1 に記載のプログラムプロダクト。

【請求項 7】 属性を作成するためのコンピュータで読取り可能なプログラムコード手段であって、ユーザによって識別される属性名から前記オブジェクトコンテキストの一部としての属性コンテキストへのマップを作成するためのプログラムコード手段と、前記属性コンテキストに前記属性に対する値を格納するためのコンピュータで読取り可能なプログラムコード手段と、を含む属性作成プログラムコード手段と、

前記マッププログラムコード手段は、前記サービスプログラムコード手段の内の少なくとも幾つかを前記属性コンテキストに結合することをさらに行う、前記ユーザによって識別される属性名を前記属性コンテキスト中の対応する前記属性の値へマッピングするコンピュータで読取り可能なプログラムコード手段と、を更に備える請求項 6 に記載のプログラムプロダクト。

【手続補正 2】

【補正対象書類名】明細書

【補正対象項目名】0005

【補正方法】変更

【補正の内容】

【0005】

【発明が解決しようとする課題】

様々に異なるシミュレーションの局面は様々な評価頻度要求を発生するので、其々の局面で最適な頻度で実行されることが望ましい。不幸にも、従来のシミュレーションは、シミュレーション全体を共通の頻度で実行させた。共通の頻度は、シミュレーションのある局面にとっては必要な頻度より多いし、他の局面にとっては理想的な頻度より少ない。例えば、もしエンジンでの熱蓄積の計算は、複雑なので一秒につき一回か二回しか実行されないならば、この場合、エンジンの部品位置の計算も遥かに簡単な演算なのに、一秒につき一回か二回しか実行されない。更に、位置が一秒につき一回か二回しか更新されないの、エンジンの周囲における動作の描画も著しく阻害される。

【手続補正 3】

【補正対象書類名】明細書

【補正対象項目名】0008

【補正方法】変更

【補正の内容】

【0008】

【課題を解決するための手段】

本発明の第 1 の形態によれば、シミュレーションを実行するためにコンピュータシステムで使用されるコンピュータプログラムプロダクトであって、コンピュータで読取り可能

な複数のサービスプログラムコード手段であって、前記コンピュータで読取り可能なサービスプログラムコード手段は、シミュレーションされる処理の下で環境のオブジェクトのシミュレーションされる属性を集散的に決定するように構成され、各サービスプログラムコード手段は、オブジェクトコンテキスト内のオブジェクト属性の少なくとも1つのサブセットに関連づけられ、前記コンピュータで読取り可能なサービスプログラムコード手段の中の少なくとも幾つかは、前記サービスプログラムコード手段による前記オブジェクトコンテキストが認識可能な前記属性の識別名に基づいて行われる前記サービスプログラムコード手段と前記オブジェクトコンテキストとの間の相互通信を行うため、および、前記オブジェクト属性に対して処理を行うため、前記サービスプログラムコード手段を前記オブジェクトコンテキスト中の前記属性に結合する、コンピュータで読取り可能な属性アクセスプログラムコード手段を備える、コンピュータで読取り可能な複数のサービスプログラムコード手段と、前記サービスプログラムコード手段のうちの少なくとも幾つかを前記オブジェクトコンテキストに結合することによって、前記オブジェクトコンテキストが認識できないユーザによって表現された属性名を前記オブジェクトコンテキストが認識可能な前記属性の識別名へマッピングするコンピュータで読取り可能なマッププログラムコード手段と、を備えることを特徴とするプログラムプロダクトが提供される。

コンピュータプログラムプロダクトは、シミュレーションを実行するためにコンピュータシステムで使用されるものである。コンピュータプログラムプロダクトは、コンピュータで読取り可能な複数のサービスプログラムコード手段を包含している。前記サービスプログラムコード手段は、シミュレーションされる処理の下で環境のオブジェクトのシミュレーションされる属性を集散的に決定するように構成される。

【手続補正4】

【補正対象書類名】明細書

【補正対象項目名】0015

【補正方法】変更

【補正の内容】

【0015】

本発明の実施例によれば、多数のサービスが非同期的に動作する。しかし、多数の非同期的に動作するサービス間のデータ通信は、間違いなく同期化される。これは、サービスの様々な非同期的な「フレーム頻度（速度）」に適応しながらも実現される。様々なサービスによって必要とされるオブジェクトデータの複数の「一覧（表示）」を支援するために、サービスによりアクセスが要求されたオブジェクト全部が「ノード」ベースのクラスから引出される。ノードはサービスと交流してデータ要素にフレーム間違いのない（フレームセーフ）アクセスを行う。各ノードは、オブジェクト（あるいはオブジェクトの一部）毎に多数の「ノード画像」を維持する。各サービスは、様々な時点で異なったセットの変更をとらえる可能性がある。ただし、サービスは全部が究極的には同一の値に作用する。一つの実施例によれば、メモリへの影響を低減するために、「ノード」と呼ばれるオブジェクト内において、比較的木目の細かい粒度でデータの複写が行われる。ノードは、一般的に、比較的小さな単位（オブジェクトの一部）であり、データのコピーは、それを必要とするノードのためだけに保持される。こうして、メモリの複写は低減される。

【手続補正5】

【補正対象書類名】明細書

【補正対象項目名】0021

【補正方法】変更

【補正の内容】

【0021】

次に（または、ステップ304の判断が「否」ならば）、ステップ308で、ノードがノード画像を保持している他のサービスに、「最新の画像」が変更されたことを通知する。ステップ310で、要求された書込み処理がそのサービスのためのノード画像に対して実行される（そのノード画像がサービスに対応する「古い」ノード画像であろうとサービ

スに対応する新たに作成されたノード画像あろうと実行される)。

【手続補正 6】

【補正対象書類名】明細書

【補正対象項目名】0022

【補正方法】変更

【補正の内容】

【0022】

通知(ステップ308)と通知されたサービスが通知にどのように応答するかについて述べる。図2に戻ると、通知は、変更要求処理ステップ204において「零(ヌル)」変更要求としてサービスによって処理される。特に、このような通知を受信すると、通知されたサービスは、通知したサービスによって変更されたノードに対する変更要求を作成する。しかし、この変更要求は「零(ヌル)」である。ノードによって表されるオブジェクトに書込まれるデータを含んでいないからである。むしろ、その変更要求は、ノード画像を必要に応じて「最新の画像」に対応した通知されたサービスに関連付けさせようという要求である。

【手続補正 7】

【補正対象書類名】明細書

【補正対象項目名】0026

【補正方法】変更

【補正の内容】

【0026】

図7において、サービスは其々異なるフレーム頻度(異なる時間周期を使って)で実行される。各サービスは別個のスレッドで実行されてもよい。図7は、ノード処理によってどのように同期の見掛けがサービスに提供されるかを示す「実時間」の時間線図である。まず最初にサービスAを取り上げる。サービスAには処理のためにノード属性が提供されている(矢印602aで示す)のが分かる(略記では、属性の「読取り」となっているが、この場合の「読取り」はノードオブジェクトを介して行われる)。その後のある時点において(矢印604aで示す)、サービスAはオブジェクト属性の書込み処理を待ち行列に入れる。更に後に(矢印606aで示す)、保護された「クリティカル」処理セクションの間に、サービスAは、適宜のノードにオブジェクト属性を画像(現在のあるいは「最新の」画像)に書込ませる(書込むよう要求する)。これは、サービスBとサービスCについても同様であり、図7においてサービスAの場合と同じ参照番号に下付き文字BとCとを用いて示す。

【手続補正 8】

【補正対象書類名】明細書

【補正対象項目名】0029

【補正方法】変更

【補正の内容】

【0029】

属性を使用しなければ、オブジェクトのハードコード化されたメンバである変数に、ユーザは、一般的には、オブジェクトによって提供される直接的な方法(経路708)すなわち、ユーザ専用のインタフェース用対話を介してのみアクセス可能である。一方、図9に示すように、属性スキーマ702は、属性名(一般的には文字列)をシミュレーションの内部の「ハードコード化された」変数へリンクする関連マップを提供する。

【手続補正 9】

【補正対象書類名】明細書

【補正対象項目名】0030

【補正方法】変更

【補正の内容】

【0030】

属性はユーザによって追加されることもある（すなわち、シミュレーションの「ハードコード化された」変数へマップされるのではなく）。この場合、属性スキーマ 7 0 2 には、属性文脈 8 0 4 内の属性を指示すポインタ値 8 0 2 が設定される。ユーザによって付加された属性はハードコード化されたものではないので、そのようなユーザによって付加された属性にアクセスするために専用のユーザインターフェイス用対話は存在しない。ユーザによって付加された属性には、包括的な属性アクセス用ユーザインターフェイス（一般的にはサービス 1 0 4 の一つとして実行される）を介してのみアクセス可能である。ユーザによって付加された属性は、一般的には、プロジェクトで使用される資源を追跡するために使用される。例えば、あるプロジェクトでは、ABS ブレーキシステムの設計にかかわる数百もの資源を含有する。ユーザは、部品名、重量、大きさ、価格、製造日時などを設定するために構成部品に属性を割当てることができる。その後、ユーザは、問合せを利用して当該プロジェクトを検索し、価格が 4 0 ドル以下で重量が 3 ポンド以下の 1 9 9 5 年以降に製造された構成部品全部を探し出すことができる。

【手続補正 1 0】

【補正対象書類名】明細書

【補正対象項目名】0 0 3 1

【補正方法】変更

【補正の内容】

【0 0 3 1】

また、属性によって、ユーザはシミュレーション中に構成部品の行動に影響を与えることが出来る。すなわち、組込み属性は、構成部品が何であり何ができるかという構成部品の行動に貢献する。例えば、三次元場面における構成部品の $x - y - z$ 位置という属性がそうである。ユーザは、構成部品に関連する属性の値を修正することによって構成部品の行動に変更を加えたり影響を与えたりできる（再び、一般的には、包括的な属性アクセスユーザインターフェイスを介して）。また、ユーザは、シミュレーション中に属性の値を監視できる。「ノード」全部（すなわち、システム内の全ての公開オブジェクト）の基本クラスは、文字列を変数の名称として使用して属性にアクセスする包括的な方法（メソッド）を提供する。処理時、属性オブジェクトは、属性の文字列定義名を対応する値が実際に格納された位置へ動的にマップする。これは、属性が「ハードコード化」された変数に対応しているか属性がユーザによって付加されたものかに係わりなく行われる。これは、コード開発者がオブジェクトのメンバ変数にアクセスする方法（ソースコード内でメンバ変数名を指定することによる）と対比をなす。すなわち、属性オブジェクトの処理が、属性の文字列定義名をオブジェクト内の値の格納位置に整合するあるレベルの抽象化を提供する。