仮想世界データベースシステム VWDB2 における仮想世界同期法

渡辺 知恵美[†] 増永 良文[††]

我々が開発を行っている仮想世界データベースシステム version2（VWDB2）では，協調作業支援の視点からデータベース機能を備えたネットワークバーチャルリアリティ（NVR）システムの実現を目指している．VWDB2 はデータベース機能によって NVR システムをサポートすることにより，NVR システムで共有された仮想世界に対し高い信頼性および同期性を保障することを目的とする．我々はまず NVR システムで共有される仮想環境の信頼性を保障するために，複数の VR システムと 1 台のバックエンドデータベースシステムによるクライアントサーバ形式のシステム構成をとり，仮想環境で行われるすべての更新操作をデータベースへのトランザクションとしてデータベースサーバで管理するためのトランザクションモデルを導入してきた[17]．本稿では，仮想環境の同期性を保障するための仮想世界同期法を提案し，その有効性を検証する．VWDB2 では移動などの連続的な操作を行う場合，一定時間ごとに更新要求を行うことによって各クライアントとサーバとの同期を行う．この更新要求を発行する間隔を縮めることによって同期性を高めることができるが，その一方でサーバへのアクセス集中がおこり，全体のパフォーマンス低下を引き起こす可能性がある．そこで，「共有ゴーストオブジェクト」という同期法を新たに導入した．共有ゴーストオブジェクトの導入によりサーバへの同期間隔にかかわらずクライアント間で一定に高い同期性を保つことができる．実験では本同期法の有効性を確認し，サーバへのアクセス集中を大幅に軽減できることを示した．

A Synchronization Mechanism for the Virtual World Database System: VWDB2

Chiemi Watanabe[†] and Yoshifumi Masunaga[††]

In this paper, the VWDB2, a network virtual reality system with a database function, is investigated particularly from the cooperative work support point of view. In order to realize the database function in the VWDB2, a set of virtual reality systems are system-integrated with a single back-end database system. A novel transaction model is introduced where three types of transactions are introduced, namely primitive transactions, group transactions, and continuous transactions. In the shared work environment provided by the VWDB2, more than one worker may issue continuous transactions concurrently. In that case, some abnormal phenomena are observed mainly due to the inconsistency of database states among virtual reality front-end systems. In order to resolve these phenomena, the neighboring ghost objects are introduced. The ghost objects are effective at eliminating the above difficulties. Based on the neighboring ghost objects, a novel synchronization model is implemented on the VWDB2 to realize a shared work environment. To verify the effectiveness of our approach, some experiments are done by using a new game named the block composition game created for this purpose. It is shown that the approach proposed in this paper ensures both high reliability and high synchronism which are known as the essential features for realizing an efficient shared work environment.

1. はじめに

近年，フライトシミュレーションや仮想美術館など様々な分野でのバーチャルリアリティ（VR）システムの実用化にともない，VR システムにおけるデータベース機能の必要性が認識されつつある．実際に，ウォークスルーにともなう検索機能[8]や仮想世界オブジェクトの永続的管理機能[4]など，VR システムの一部の機能をデータベース機能によって管理したり機能拡張したりしたシステムも開発されてきている．我々は，VR システムはデータベース機能を完全に備え持つことによってより強力なシステムとなると考え，VR システムと DB システムをシステム連携した Virtual

† 奈良女子大学大学院人間文化研究科
Graduate School of Humanity and Science, Nara Women's University
†† お茶の水女子大学理学部
Faculty of Science, Ochanomizu University

World Database System（VWDB）の開発を進めてきた[9),10)]．VWDB は当初 1 台の VR システムと 1 台のデータベースシステムをシステム連携してプロトタイピングが行われたが，最新のバージョンである VWDB Version 2（VWDB2）では複数台の VR システムと 1 台のデータベースシステムを統合してプロトタイプが構築されている．そのような構成の VWDB2 は，仮想共同作業環境（Shared work environment）の支援に有効に働くと考えられる．ここに，仮想共同作業環境とは，同じ世界に没入したユーザが共有化された仮想世界オブジェクトを共同で編集できる環境であり，複数の VR システムをネットワークで接続したネットワーク VR（NVR）システム[11)]によって実現される．我々は VWDB2 に仮想共同作業環境を実装し，仮想共同作業環境をサポートするためのデータベース機能を提供している．まず，文献 17) において仮想共同作業環境で必要とされる信頼性を保障するためにクライアントサーバ形式をとり，仮想共同作業環境で行われるすべての更新操作をデータベースへのトランザクションとして管理するためのトランザクションモデルを導入した．本稿では，仮想共同作業環境の同期性を保障するための仮想世界同期法を提案し，その有効性を検証する．VWDB2 では移動などの連続的な操作を行う場合，一定時間ごとに位置更新の更新要求を行うことによって各クライアントとサーバとの同期を行う．この更新要求を発行する間隔を縮めることによって同期性を高めることができるが，その一方でサーバへのアクセス集中がおこり，全体のパフォーマンス低下を引き起こす可能性がある．この問題を解決するために我々は「共有ゴーストオブジェクト」による仮想世界同期法を新たに導入した．共有ゴーストオブジェクトの導入により，仮想共同作業環境における同期の役割を 2 種類に分離して表示することができるため，サーバへの同期間隔にかかわらずクライアント間で一定に高い同期性を保つことができ，その結果サーバへのアクセス集中を軽減することができる．本稿の構成は以下のとおりである．まず 2 章 VWDB2 について，3 章にて文献 17) で導入した VWDB トランザクションモデルについてその概要を述べる．そして 4 章で今回導入した共有ゴーストオブジェクトによる仮想世界同期法について述べ，その有効性の検証の結果を 5 章で示し，6 章で結論を述べる．

2. 仮想世界データベースシステム：VWDB

2.1 システム概要

データベース機能を完全に VR システムに組み込むことを我々はバーチャルリアリティをデータベースにする（Making Virtual World a Database System）と呼ぶ．VWDB の目的はまさに「バーチャルリアリティをデータベースに」したシステムの実現であり，VWDB データモデルの設計とプロトタイプシステムの実装を進めている．プロトタイプシステムは，バックエンドのオブジェクト指向データベースシステムと複数台の VR クライアントとのサーバ–クライアント形式の構成で開発している（**図 1**）．VR クライアントは JAVA 言語用の仮想世界構築 API である JAVA3D を用いて実装し，バックエンド DB は商用のオブジェクト指向 DB であるエクセロン社の ObjectStore を利用し JAVA 言語で実装している．

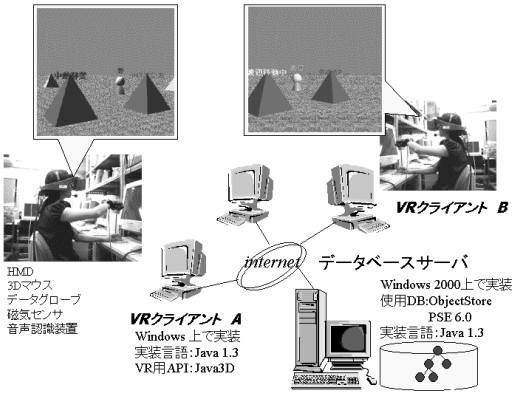

図 1 システム構成
Fig. 1 System architecture of the VWDB2.

VWDB2 では複数のクライアントが同じデータベースサーバに接続することによって仮想共同作業環境を実現する．これを実現するためのネットワークトポロジとして，我々はデータベースサーバを中心としたマルチキャストによる集中管理型接続方式[2)]を採用した．クライアントとサーバ間の通信プログラムには HORB[7)]を用い，マルチキャストは HORB extention メッセージングサービスを用いた．VWDB で VR クライアントのアプリケーションを起動すると，VR クライアントはデータベース中にある仮想世界のコピーをローカルに作成し，VR システム上に表示する．そして，各 VR クライアントの持つ仮想世界とデータベース中に格納されている仮想世界との同期をとることですべてのクライアントが「同じ仮想世界を共有している」感覚で VR システムを利用することができる．**図 2** に VR クライアント A が更新を行ったときの同期処理の流れを示す．VR クライアント A が仮想世界オブジェクトの操作を実行すると，同じ操作がデータベースサーバに送られ（図 2 (1)），データベー

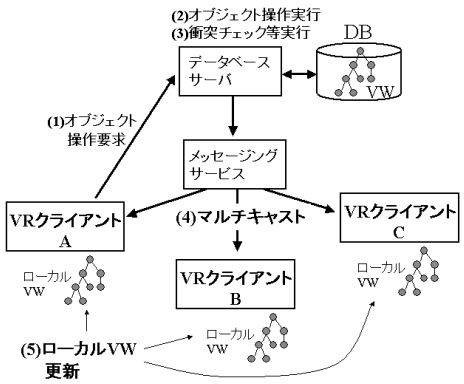

図 2 マルチキャストによる仮想共同作業環境の実現
Fig. 2 Update synchronization of the VWDB2 clients by multi-cast.

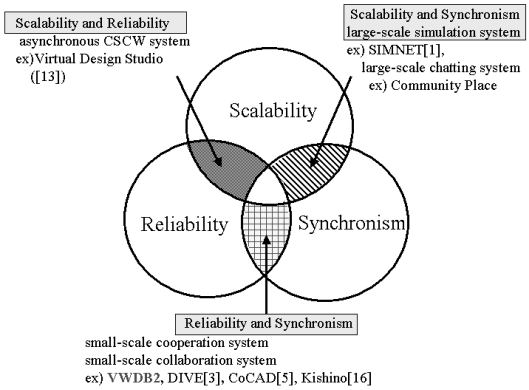

図 3 NVR システムに要求される 3 要素
Fig. 3 Three factors required for the network virtual reality system.

スサーバはこれを実行する（図 2 (2)）．このとき仮想世界におけるオブジェクトどうしの衝突チェックや物理法則などを実行し（図 2 (3)），メッセージングサービスによって他 VR クライアントにマルチキャストする（図 2 (4)）．マルチキャストされたデータは，各 VR クライアントがローカルに持つ仮想世界を更新する（図 2 (5)）．これにより，データベース中にある仮想世界と各 VR クライアントの仮想世界との同期をとることができる．

2.2 VWDB2 の特徴：信頼性と同期性

VWDB2 は NVR システムで実現される仮想共同作業環境の「信頼性」と「同期性」をより強力にサポートするところに特徴がある．本節ではこの特徴を従来の NVR システムとの比較により明らかにする．

NVR システムは，1983 年に開発された SIMNET [1] をはじめとしてこれまでに多くのシステムが開発されてきた．これらのシステムでは，「スケーラビリティ（Scalability）」，「同期性（Synchronism）」，「信頼性（Reliability）」の 3 要素を満たすものが理想とされる．スケーラビリティとは同時に利用できるユーザ数の多さ，同期性はクライアント間での仮想世界の同期の頻度を表し，信頼性は仮想共同作業環境におけるトランザクションの ACID 特性が厳密に保障されるほど高い．NVR システムはこれらの 3 要素をすべて満たすのが理想的であるが，これらの要素はすべてトレードオフの関係にある [11] ため，すべての要素を満たすのは難しい．そのため，NVR システムはこれらの要素のうち，2 要素を満たすよう設計されている．**図 3** はこの視点に基づいて NVR システムを分類したものである．図 3 に示されるように，スケーラビリティと同期性を重視する NVR システムは，数百人から 1,000 人規模によるネットワークゲームや軍事シミュレーション，チャットシステムなどを対象としているものが多い．しかしこれらのシステムはユーザが行う更新を永続的に保存する必要がない．つまりゲームやシミュレーション，チャットシステムは 1 回のセッションが終了した場合仮想世界をその状態に保つ必要はなく，かえってセッション前の状態に戻すことが多い．そのため，ログをとる以外はユーザの更新によって変更される仮想世界をリアルタイムにデータベースで保存し管理する必要がなく，仮想共同作業環境における一貫性は保障する必要があるが，持続性（Durability）および原子性を保障する必要がない．

一方，共同 3DCAD システムなどは，ユーザの更新によって変更される仮想世界および仮想世界オブジェクトをリアルタイムに保存し永続化する必要があり，仮想共同作業環境における一貫性，分離性はもちろんのこと原子性，持続性を保障する必要がある．香港大学とワイマール大学とで試みられた Virtual Design Studio [13] では，信頼性とスケーラビリティを重視し，非同期的な NVR システムを用いてヘリポートの共同設計を行っている．このシステムでは仮想世界の更新を行うユーザを 1 人のみで行うように限定しており，他のユーザは，ユーザどうしのチャットおよびウォークスルーはできるものの更新はできないようになっている．しかしながら VWDB2 では複数ユーザが同時に操作を行うことのできる環境を対象としているため，同期性が重要である．そこで VWDB2 ではスケーラビリティを 2 人から数十人と限定する代わりに，仮想共同作業環境の信頼性と同期性を確保することを目的とする．DIVE [3]，CoCAD [5]，Kishino [16] も同じく信頼性と同期性を重視しているが，そのためのデータ

ベース機能は十分に備わっていない．

以上の特徴を考慮し，我々は，VWDB2 の実現する仮想共同作業環境で実現される代表的なアプリケーションとして共同作業による共同 3D-CAD システムを考えている．遠隔地にいるユーザどうしが同じ仮想共同作業環境に没入し，会話をしながらオブジェクトをつかんでレイアウトしたりお互いの持つオブジェクトを組み合わせて建築物のレイアウトを行ったりする．互いの持つオブジェクトを合成したり対応関係を考えてレイアウトしたりするには相手のオブジェクトの位置をリアルタイムに知る同期性の高さが必要であり，VWDB2 はこれを保障する．また，仮想共同作業環境におけるユーザの操作によって設計された仮想世界および仮想世界オブジェクトはすぐさま共同設計の作品としてデータベースにて永続化され，一貫性や分離性だけでなく，操作の原子性や持続性が保たれた状態で他のユーザと共有される．これらを実現するために VWDB2 は VWDB トランザクションを導入した[17]ことによって仮想共同作業環境の信頼性を保障する．このような共同 3D-CAD システムでは共同して操作を行うユーザ数は限られており，スケーラビリティは数人から数十人程度で十分であると考えられる．なお，ネットワークゲームやシミュレーションシステムなどはユーザの行う操作をすぐさま永続的に繁栄する必要性があまりないことや大規模なスケーラビリティが必要であることから VWDB2 の提供する仮想共同作業環境は有用ではないと思われる．

3. VWDB トランザクションモデル

2.2 節の考えに基づき，我々はまず文献 17) で VWDB2 上に実現される仮想共同作業環境の信頼性を保障するための VWDB トランザクションモデルの定義を行った．VWDB は DB システムなので，それが 提供する仮想世界での問合せや更新要求をトランザクションとしてとらえることにより，様々な障害に遭遇したときに（たとえばシステムがダウンしたとき），データベースが矛盾した状態にならないことを保障したり，同一の仮想空間に複数のユーザが没入して作業を行っている場合に，矛盾のない同時実行を保障したりできる．VWDB トランザクションには，以下の 3 種類のトランザクションがある．

(1) 基本操作トランザクション
(2) グループ操作トランザクション
(3) 連続操作トランザクション

基本操作トランザクションは，オブジェクトの生成，消去，属性値の変更などの基本的な操作を表す．また

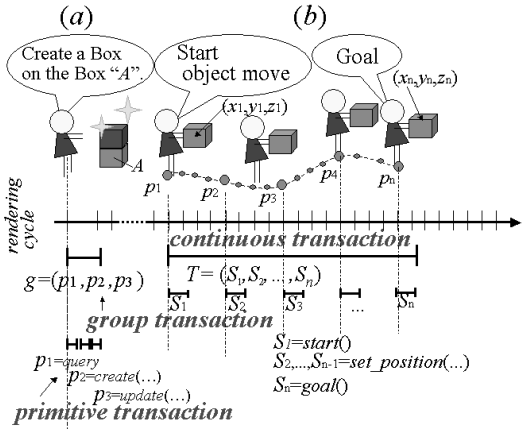

図 4 仮想共同作業環境におけるトランザクション
Fig. 4 Transactions in the shared work environment of the VWDB2.

たグループ操作トランザクションは基本操作トランザクションの組合せによるトランザクションであり，**図 4** (a) のように，query，create，update という 3 つの基本操作トランザクションの組合せで「A というボックスの上にボックスを 1 つ生成させる」というグループトランザクションを発行することができる．

また，複数回の基本操作またはグループ操作を実行することによって，たとえば移動のような意味のある一連の操作が実現されるとき，これを連続操作トランザクションとしてまとめることができる．連続操作トランザクションには，移動や回転，変形などがある．基本操作トランザクションとグループ操作トランザクションには従来知られているトランザクションモデルを適用する．しかし，連続操作トランザクションは，トランザクションが終了するまでに長時間を要すること，図 4 (b) の移動操作で示されるように，「Goal」とユーザが指示するまでいつ終了するか分からないインクリメンタルなトランザクションであることから，VWDB 独自の新たな概念を取り入れる必要がある．図 4 (b) に示したユーザが箱をつかんで（目的地まで）移動するという，オブジェクト移動トランザクションを考える．移動トランザクション連続操作トランザクションに分類され，箱を現在置かれている位置から目的地まで移動することをアプリケーションレベルからして 1 つの意味のある仕事の単位と考える．そこで，この箱の移動操作の開始から終了までを移動トランザクション T とし，トップトランザクション T を構成する基本操作およびグループ操作のサブトランザクションの系列 $T = (S_1, ..., S_n)$ でそれを定義する．

この例では，まず移動開始時にサブトランザクショ

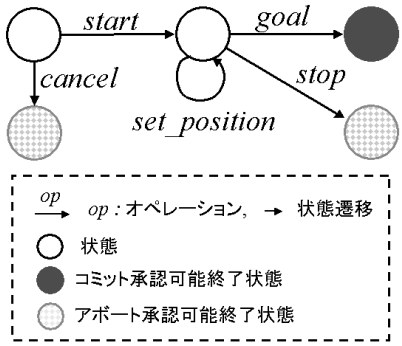

図 5 オブジェクト移動トランザクションにおける状態遷移
Fig. 5 State transition of the VWDB2 continuous transaction.

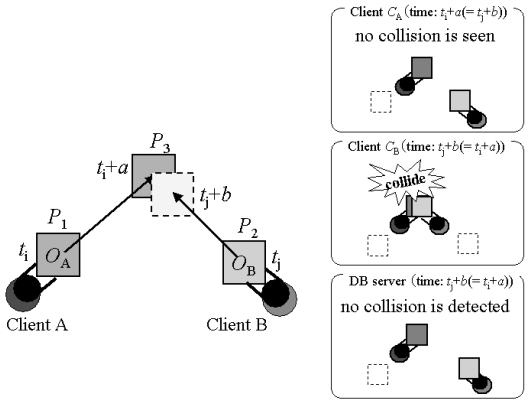

図 6 現象 2
Fig. 6 Explanation of phenomenon 2.

ン S_1 で $start()$ が実行されたのちサブトランザクション $S_i(1 < i < n)$ で $set\_position(x_i, y_i, z_i)$ が実行される．そしてユーザがオブジェクトを手離すとサブトランザクション S_n で $goal()$ を実行し T をコミットする．しかし，システム障害などで移動が中断された場合，箱が本来想定した目的地まで移動されなかったという意味では移動トランザクションはアボートされるべきであるが，VR システムにおける移動の場合，時間の非可逆性から開始からその地点までの箱の移動は完結したと見なされるべきであるから，連続操作のトランザクションでは承認終了状態という概念を導入して問題解決を図った．承認可能終了状態とは，ワークフローのためのトランザクション拡張の 1 つとして導入された[12]概念であり，トランザクションの状態と状態遷移を考えて承認可能終了状態を定めておく．承認可能終了状態には，コミット承認可能終了状態（committed acceptable termination state）とアボート承認可能終了状態（aborted acceptable termination state）があり，トランザクションの実行途中でアボートされた場合でもアボートせず，アボート承認可能終了状態でトランザクションを終了する．移動トランザクションの状態遷移を**図 5** に示す．無事に目的地にオブジェクトを移動できた場合は，トランザクションはコミット承認可能終了状態となる．移動が途中で中断された場合は，システム側で強制的に $stop()$ 操作を発行することによってその場にストップさせることでアボート承認可能終了状態とし，連続操作を終了する．

4. 仮想世界同期法

VWDB2 では連続操作トランザクションのサブトランザクションがコミットされた時点で他クライアントへトランザクションの結果がマルチキャストされることによって同期が行われる．すなわち同期間隔はサブトランザクションのことを表し，同期間隔が短くなるほど同期性が高い．しかし，同期間隔が短くなるにつれて多くのアクセスがバックエンドデータベースへ集中し，全体のパフォーマンスが低下するという問題がある．VWDB2 は同時作業人数が 2 人〜数十人という比較的小規模なアプリケーションを対象としているが，複数ユーザが同時にオブジェクトを移動した場合データベースサーバへのアクセスが集中することになる．そこで本章ではこの問題を解決するために導入した「共有ゴーストオブジェクト」による仮想世界同期法について述べる．この同期法は同期間隔が長くなる際に生じる問題点を共有ゴーストオブジェクトの導入によって解決する．これにより，ある程度同期間隔が長い場合にも高い同期性を維持することができ，サーバへのアクセス負荷を軽減することができる．

4.1 具体的な問題

本節ではまず同期間隔が長い場合に生じる問題点について具体的な例をあげて説明する．**図 6** は，仮想共同作業空間上でクライアント A とクライアント B が各々同時にオブジェクト O_A とオブジェクト O_B を移動するトランザクション $T_A = (S_1^A, \cdots, S_n^A)$, $T_B = (S_1^B, \cdots, S_m^B)$ を実行している様子である．各トランザクション T_A, T_B では発行間隔 a でサブトランザクションが発行される．このとき発行間隔が広がることによって以下のような現象が起きる場合がある．

現象 1： T_A の位置更新サブトランザクションはある時間ごとに発行され，他のクライアントにマルチキャストされるため，クライアント C_B 上ではオブジェクト O_A が急に動くように感じられる．

現象 2： オブジェクト間の衝突検出はバックエ

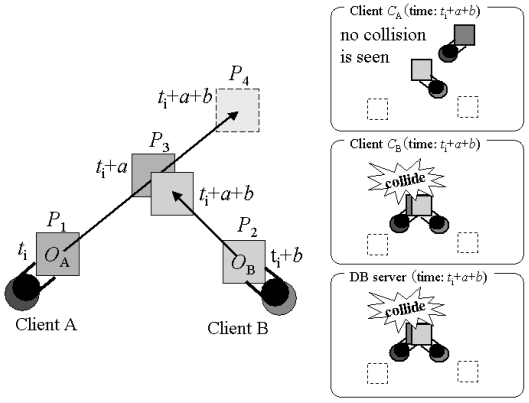

図 7 現象 3
Fig. 7 Explanation of phenomenon 3.

ンドデータベースで行われるため，クライアント側で起こったはずの衝突が検出されない場合がある．(図 6)

現象 3： 現象 2 と同じ理由で，クライアント側では起こっていないはずの衝突がバックエンドデータベース側で検出されてしまう場合がある (**図 7**)．

ここでは特に現象 2 と現象 3 を図 6 と図 7 を用いて説明する．まず，現象 2 では，ユーザ A はオブジェクト O_A を移動し，時刻 t_i に地点 P_1 で S_i^A を発行し，時刻 $t_i + a$ に地点 P_3 で S_{i+1}^A を発行している．またユーザ B はオブジェクト O_B を移動し時刻 t_j に地点 P_2 で S_j^B を発行している．ユーザ B はさらに移動を続け，時刻 $t_j + b (b < a, t_i + a = t_j + b)$ には地点 P_3 まで移動させている．この時点でクライアント B 上の仮想世界では，オブジェクト O_A と O_B が衝突しているように見える．しかし，この時刻では，オブジェクト O_B のサブトランザクションは発行されておらず，データベース上では衝突していないため，そのまま O_B はオブジェクト O_A を通り抜ける結果となる．また現象 3 では，ユーザ A はオブジェクト O_A を図 6 の右上方向に移動し，時刻 t_i に地点 P_1 への位置更新サブトランザクション S_i^A を，さらに時刻 $t_i + a$ に地点 P_3 への位置更新サブトランザクション S_{i+1}^A を発行している．そのあと，A は O_A を移動しつづけ，時刻 $t_1 + a + b (b < a)$ には地点 P_4 まで持っていったとする．一方，ユーザ B はオブジェクト O_B を図 7 の左上方向に移動し，時刻 $t_i + b$ に地点 P_2 への位置更新サブトランザクション S_i^B を，さらに時刻 $t_i + a + b$ に地点 P_3 への位置更新サブトランザクション S_i^B を発行している．このとき，クライアント C_A ではユーザ A が O_A を P_4 まで移動させていたのにもかかわらず，地点 P_3 での衝突計算が行われ，ユーザ A が意図しない方向に移動してしまう．

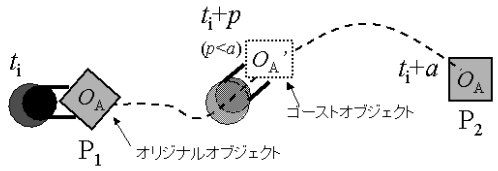

図 8 ゴーストオブジェクト
Fig. 8 Ghost object.

4.2 ゴースト オブジェクト の導入

前節の現象は，各連続操作トランザクションのサブトランザクションが一定間隔（前節では a）ごとに発行されることによって起こる．つまりサブトランザクションが発行された時刻 t から次に発行される時刻 $t+a$ までクライアント側のオブジェクトの状態とデータベース側のオブジェクトの状態が同期されていないことが原因となる．この矛盾をなくすためにどちらかの状態に統一する方法もある．しかし，各クライアントでは描画サイクルごと（約 50 ms）にローカル操作が実行されるため，データベースへ発行するサブトランザクションの間隔をそこまで狭めるのには限界がある．また，データベースの状態とクライアントの状態を整合させるためにサブトランザクションを発行させるときのみローカル操作を許すと，対話性が極端に落ちてしまう．そこで，我々は各クライアントのアプリケーションに「ゴーストオブジェクト」を導入することによってその解決を図った．

4.2.1 ゴースト オブジェクト

クライアントで連続操作トランザクションを実行しているとき，サブトランザクション間でローカルでのオブジェクトの状態とデータベース上でのオブジェクトの状態とが異なっている場合，これらを両方表示することとした．この場合，データベース上でのオブジェクトの状態を表示したものを「オリジナルオブジェクト」，クライアント上のオブジェクトの状態を表示したものを「ゴーストオブジェクト」とする．移動トランザクションにおけるゴーストオブジェクトの例を**図 8** に示す．VR クライアントでの描画サイクルを b ミリ秒間隔とし，サブトランザクション発行間隔を a ミリ秒とする．また，クライアント上で行われる処理の時間は $p (p < b)$，仮想共同作業環境における操作の処理時間は $q (b < q < a)$ であるとする．この場合，オブジェクト O_A の移動トランザクション T_A を開始すると，ゴーストオブジェクト O'_A が生成される．ユーザは基本的にこのゴーストオブジェクト O'_A を移動させることになる．時刻 t_0 に開始サブトランザ

クション S_1 が発行された後，位置更新サブトランザクション S_2 が実行されるまでの間，b ミリ秒間隔でゴーストオブジェクト O'_A の位置更新操作が実行される．これは，VR クライアントのみで完結する操作である．そして，時刻 t_0+a に S_2 が実行され，ゴーストオブジェクトの位置である P_2 地点にオリジナルオブジェクトを移動させるよう位置更新サブトランザクションを発行する．これにより，オリジナルオブジェクトは時刻 t_0+a+q に P_2 地点に移動する．ゴーストオブジェクトとオリジナルオブジェクトは線でつながれ，ゴーストオブジェクトを使ってオリジナルオブジェクトを引っ張る感覚で移動を行うことができる．

4.2.2 共有ゴーストオブジェクト

ゴーストオブジェクトを用いて 4.1 節であげた現象を緩和させるために，我々は近隣で連続トランザクションを実行しているクライアントどうしが互いのゴーストオブジェクトを共有させる「共有ゴーストオブジェクト」を導入する．ここで「オブジェクト O_A が O_B の近隣にある」とは，仮想共同作業環境におけるオブジェクト O_A の位置座標が O_B の位置座標からある閾値以内の距離に存在することをさす．たとえば，クライアント C_A，C_B，C_C から各々ユーザ A，B，C がログインしてオブジェクト O_A, O_B, O_C の移動トランザクションを実行し，現在，データベース上の仮想世界が**図 9** (a) に示す状態になっているとする．このとき，次のサブトランザクションまでの移動可能距離を r とし，操作中のオブジェクトを中心とした半径 r の球状の範囲をオブジェクトの「近隣」であると考える．図 9 (a) では O_A と O_B，O_A と O_C がそれぞれ近隣となる．このとき，O_A の移動トランザクション T_A を発行しているクライアント C_A と，O_B の移動トランザクション T_B を発行しているクライアント C_B との間に P2P のネットワーク通信路を確保し，O_A のゴーストオブジェクト O'_A と O_B のゴーストオブジェクト O'_B を共有する（図 9 (b)）．ゴーストオブジェクトどうしの同期は近隣にいるゴーストオブジェクトどうしでのみ行われることから，クライアント–サーバ間の通信で生じるようなアクセス集中がおこる可能性は少なく，クライアント–サーバの同期間隔と比べてかなり短い間隔で同期を行うことが可能である．なお，P2P 通信で共有するゴーストオブジェクトはあくまでオリジナルオブジェクトに対する仮の姿なので，オリジナルオブジェクトに比べて一貫性を厳密に確保する必要性は低いため，楽観的な一貫性管理手法[15]を取り入れる．近隣の範囲は広げすぎると P2P 通信を行うユーザ数が増えクライアントの負担が

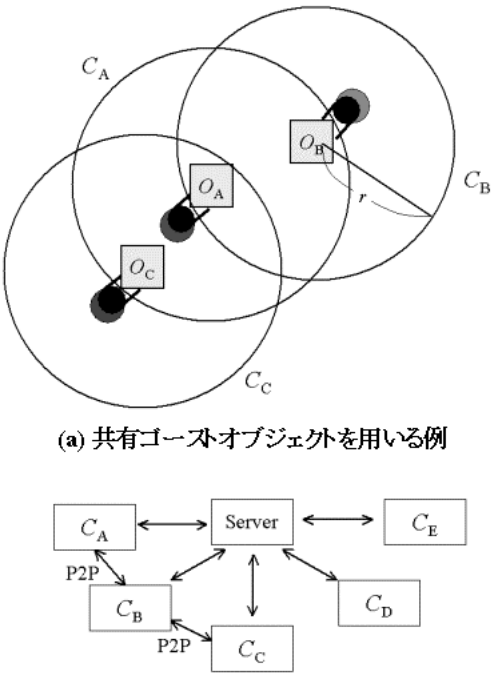

図 9 共有ゴーストオブジェクト
Fig. 9 Neighboring ghost objects and P2P connections.

大きくなり，また狭くしすぎると近隣のオブジェクトが認識しにくくなったり急に目の前にオブジェクトが現れたりするような現象が起きてしまうため，適切な範囲を選ぶ必要がある．適切な範囲はアプリケーションによって大きく異なると考えられ，サブトランザクションの発行間隔やユーザの移動距離などが関係するので，現段階ではアプリケーション設計者が範囲の設定を行うようにしている．

また，ゴーストオブジェクトを共有している最中に障害が起きた場合の対処法についてであるが，クライアント C_A と C_B がゴーストオブジェクトを共有している場合，起こりうる障害には以下の 2 種類があげられる．

(1) C_A または C_B で実行中のトランザクションがアボートする．
(2) C_A と C_B の P2P 接続が途切れる．

(1) が起きた場合，オリジナルオブジェクトは 3 章で記述した VWDB トランザクションモデルに従い，アボート直前にコミットされたサブトランザクションまでが有効となる．しかし，ゴーストオブジェクトはあくまで非永続的なオブジェクトであることからアボート直前に表示されていたゴーストオブジェクトの位置は有効にならず破棄される．また (2) が起きた場合は

再接続する際にサーバ側に通知を行って C_A と C_B が操作しているオブジェクトが近隣にあるかどうかを再確認し，近隣にあればゴーストオブジェクトの共有を再開する．

4.2.3 共有ゴースト オブジェクト 導入の意義

ここで，本稿で提案した仮想世界同期法における共有ゴーストオブジェクトの意義を明確にする．同期性とは仮想共同作業環境における同期の頻度であり，仮想共同作業環境における同期は以下の 2 種類に分けられる．

(1) **クライアント 間の同期**
複数の VR クライアント間での仮想世界の同期の頻度が高いほど，クライアント間の同期性が高くなる．仮想共同作業環境ではクライアント間の同期性が高いほど他のユーザの行っている操作や仮想共同作業環境の最新の状況をリアルタイムに確認することができ，共同作業をスムーズに行うことができる．ネットワーク VR における同期性は通常こちらの意味で表現される．

(2) **クライアント –サーバ間の同期**
VR クライアント上に表示される仮想世界と，バックエンド DB 内に格納されている仮想世界との同期の頻度が高いほどクライアント–サーバ間の同期性は高くなる．クライアントから発行されるトランザクションをサーバ側で処理しコミットすることによって同期が行われるため，クライアント–サーバ間の同期性が高いということは，クライアントで行われる操作が即座にデータベースでコミットされており，クライアント上に表示される仮想世界の信頼性が高い状態であるといえる．

つまり，クライアント間の同期はユーザどうしの共同作業をスムーズにする役割を持ち，クライアント–サーバ間の同期はクライアント上の仮想世界の信頼性を高める．今回導入した共有ゴーストオブジェクトによる仮想世界同期法はこれら 2 種類の同期の役割を分離させ，また双方を表示することによって各々の同期の役割をユーザに意識させることに特徴がある．クライアント間の同期を共有ゴーストオブジェクトで表し，近隣のユーザ間での共同作業を行う際ユーザは共有ゴーストオブジェクトを視認する．またクライアント–サーバ間の同期をオリジナルオブジェクトで表し，ユーザは定期的にオリジナルオブジェクトが更新される様子を見ることによって，自らの行ったトランザクションがコミットされたこと，現在のオリジナルオブジェクトのある位置が一貫性を持つ仮想世界オブジェクトの状態であること，またこの状態が他のすべてのクライアントで表示されていることを確認することができる．そして，たとえばオブジェクトどうしの部品合成作業などの細かい共同作業が要求されるときに必要なクライアント間の同期性は共有ゴーストオブジェクトの同期間隔にのみ依存し，クライアント上の仮想世界の信頼性を高めるために必要なサーバ–クライアント間の同期性は連続操作トランザクションにおけるサブトランザクション発行間隔にのみ依存するといったように，同期間隔の長さも役割にあわせて分離することができる．本稿の最初にあげた「高い同期性を要求するアプリケーション」とは，たとえばオブジェクトどうしの部品合成作業などの細かい共同作業が要求されるアプリケーションをさし，このような場合に要求されるのはクライアント間の同期性である．本同期法の場合，クライアント間の同期性は共有ゴーストオブジェクト間の同期間隔にのみ依存する．共有ゴーストオブジェクト間の同期間隔はスケーラビリティに関係なく一定に保つことができるため，結果「高い同期性を要求するアプリケーション」に対応することができる．

4.2.4 ゴースト オブジェクト の共有による解決策

前項で述べたようにゴーストオブジェクトを近隣にいるクライアントどうしで共有することによって，4.1 節であげた現象を最小限に抑えることができる．まず現象 1 は，近隣オブジェクトのゴーストを他のクライアントが見られるようにすることで，次のサブトランザクションで更新されるオブジェクトの状態を予測することができるため，解消される．現象 2 は以下のように解消する：オブジェクト O_A と O_B が近隣にあり，C_A と C_B が P2P で接続され，ゴーストオブジェクト O'_A と O'_B が共有されているとする．

ゴーストオブジェクト O'_A と O'_B が衝突したときの制御フローを**図 10** に示す．C_A と C_B の移動トランザクション T_A と T_B は，通常では一定間隔（ここでは a）でサブトランザクションを発行するが，ゴーストオブジェクト O'_A と O'_B の衝突がクライアント上で検出された時点（C_B は O'_B を更新したとき，C_A は O'_B 更新を受信したとき）で，特別に各々サブトランザクションを発行する．これにより，バックエンドデータベース側でオリジナルオブジェクト O_A と O_B の衝突が検出されることとなる．また，各クライアントではサブトランザクション発行後，ゴーストオブジェクトどうしの衝突計算を行い，各々ゴーストオブジェクト O'_A と O'_B の位置を更新する．その結果，各クライアントでは，共有されたゴーストオブジェクトの衝突，およびオリジナルオブジェクトの衝突が実

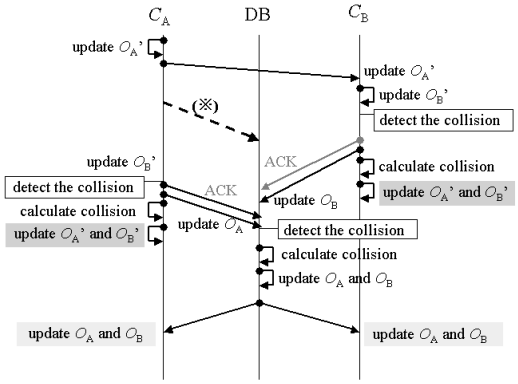

図 10 共有ゴーストオブジェクトによる現象 2 の解決
Fig. 10 Resolution of phenomenon 2 using neighboring ghost objects.

行されることになり，現象 2 は解決される．また現象 3 については，サーバ側で衝突が検出されたときにクライアントでの衝突検出の通知（図 10 の ACK）がない場合はその衝突を無効にすることによって解決することができる．なお，たとえば図 10 の場合 CB が衝突を検出した後に図中の（※）のように定期更新が行われて衝突が検出された場合でもその衝突は無視されるため，最初にゴーストオブジェクトどうしでの衝突が検出された状態で一貫して衝突後の処理が行われることになる．

5. 共有ゴースト オブジェクト の評価

5.1 評価用アプリケーション

今回提案したゴーストオブジェクトの有効性を評価するために，これまで構築してきた VWDB2 プロトタイプシステム上に次のアプリケーションを考案し，実装した．**図 11** (a) はアプリケーションのスクリーンイメージである．仮想世界は図 11 (b) に示すように 4 タイプのブロックが 2 個ずつ計 8 個並べられている．同じタイプの 2 つのブロックは合成すると 1 つの立方体となるようになっている．この世界に 2 人のユーザが没入し，図 11 (b) に示すように，2 列に並べられたブロックをはさむようにして向かい合う．そして，同じタイプのブロックを持ち，これらを合成していく．ただし，合成する際両方のブロックは誰かによって移動中でなければならない．静止したブロックにもう一方のブロックを合わせても合成することはできない．このようにして最右の青いブロックから順にブロックを合成し 4 タイプすべてのブロックを合成できたら作業完了となる．この評価用アプリケーションでは，仮想共同作業環境で 2 つ 1 組となるオブジェクトをピタリと合成させるという細かい作業を行う際に，お互いのオブジェクトのリアルタイムな状況を知る必要があり，仮想共同作業環境での同期性が重要となると同時に 2.2 節で述べたような共同 3D-CAD システムにおける作業の一例として考えることができる．

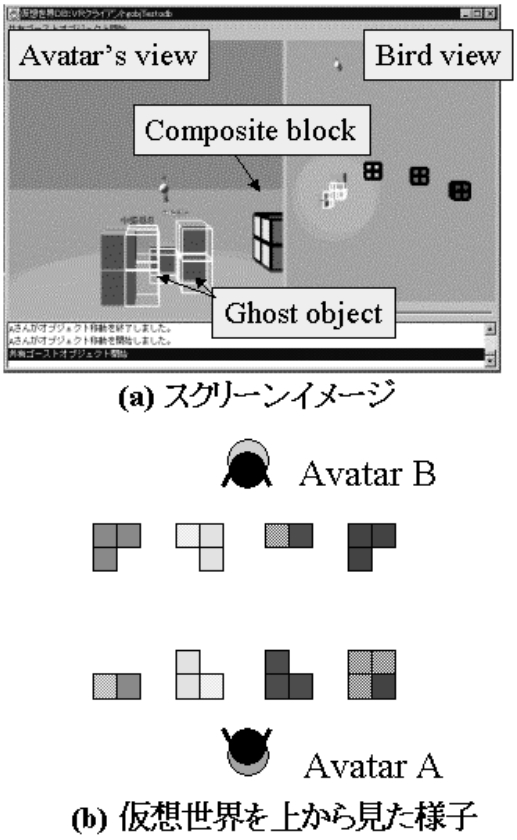

図 11 評価用アプリケーション
Fig. 11 Block composite game. (a) screenshot, (b) bird eye's view of the virtual world.

5.2 有効性の評価

前節で述べたアプリケーションを用いて今回提案した共有ゴーストオブジェクトの有効性の評価実験を行った．実験方法は以下のとおりである：5.1 節で述べた作業をゴーストオブジェクトを使用すると場合としない場合，かつ 4 種類のクライアント–サーバ間の同期間隔（1,000 ミリ秒，3,000 ミリ秒，5,000 ミリ秒，8,000 ミリ秒）の計 8 回行い，作業が完了するまでの時間を計測した．また各参加者に「他のブロックとスムーズに合成させることができたか」について**表 1** に表す 5 段階評価でアンケートをとった．参加者は 20 人で，2 人 1 組で 10 グループ（チーム A J）を組んでこれらの作業を試行した．なお，作業時間には慣れの問題が影響すると予想されたため，作業者 20 人と

表 1 主観的評価基準
Table 1 Subjectivity evaluation basis.

ランク	評価
5	とてもスムーズにすべてのブロックを合成させることができた.
4	それほど苦労せずにすべてのブロックを合成させることができた.
3	時々困難を生じるときがあったが，すべてのブロックを合成させることができた.
2	ブロックを合成させるのが困難であった
1	すべてのブロックを合成させることができなかった

もにゴーストオブジェクトを使用する場合と使用しない場合とで 1 回ずつ練習をしたのち実験を行った．また同期間隔をユーザに告げず，計 8 回の試行をランダムに実行した．この実験では，データベースサーバに PC（Pentium III 850 MHz，Windows 2000）を，クライアントには PC（Pentium III 700 MHz，Windows 2000：すべて同スペック）を利用し，Ethernet（100BASE-T）で接続した．共有オブジェクトどうしの通信間隔は一定とし 200 ミリ秒とした．また，共有ゴーストオブジェクトを使用する場合は「近隣」と見なされる範囲を「オブジェクトの中心座標から半径 7 メートル以内」とし，範囲の広さを固定した．各試行におけるアンケートの結果を**表 2** に，各試行にかかった時間を**表 3** に示す．この結果を用いて，我々は以下の観点から共有ゴーストオブジェクトの有効性に関する考察を行った．

考察 1：共同作業における共有ゴースト オブジェクトの有効性

まず共同作業の際にユーザにとって共有ゴーストオブジェクトが有効に機能したかについて評価を行う．前章にも述べたように，共有ゴーストオブジェクトの特徴はクライアント間の同期とクライアント–サーバ時の同期を分担することでその役割分担を明確にすることにある．クライアント間の同期間隔は一定に 200 ミリ秒であることから，オブジェクトの合成作業の際，ユーザがゴーストオブジェクトの役割を理解し共有ゴーストオブジェクトを視認しながら作業をすれば，サーバ–クライアント間の P2P による同期間隔の長さにかかわらず作業時間および評価は一定になるはずである．表 2 を見ると，共有ゴーストオブジェクトを使用した場合の評価点は 3.50 から 4.57 であり，ゴーストオブジェクトを使用しない場合の評価点である 2.71 から 4.43 より全体的に評価が高いことが分かる．またゴーストオブジェクトを使用しない場合は，サーバ–クライアント間の同期間隔が長くなるほど評価が悪くなるのに対し，ゴーストオブジェクトを使用した場合は

表 2 アンケート結果
Table 2 Score by 20 players for each trial.

Ghostobject	使用する				使用しない			
同期間隔 (ms)	1000	3000	5000	8000	1000	3000	5000	8000
a(TeamA)	3	4	4	4	4	2	3	2
b(TeamA)	4	4	5	4	4	3	3	3
c(TeamB)	4	5	5	5	4	3	3	3
d(TeamB)	4	4	4	4	4	3	3	2
e(TeamC)	5	5	5	4	4	3	3	2
f(TeamC)	5	5	5	4	4	3	3	2
g(TeamD)	4	5	5	4	3	3	3	2
h(TeamD)	5	4	5	4	5	3	4	4
i(TeamE)	5	5	4	4	5	5	3	4
j(TeamE)	3	5	5	3	5	4	4	2
k(TeamF)	5	3	5	4	5	3	4	4
l(TeamF)	4	3	2	4	4	4	3	2
m(TeamG)	4	4	5	3	4	4	3	2
n(TeamG)	3	5	5	3	4	4	4	3
o(TeamH)	3	3	4	3	2	2	2	2
p(TeamH)	3	4	5	3	4	3	2	2
q(TeamF)	5	4	5	4	4	4	2	2
r(TeamI)	4	4	4	4	5	3	3	3
s(TeamI)	4	4	5	3	5	3	3	3
t(TeamJ)	4	5	5	3	5	4	3	3

表 3 ゲーム終了までにかかった時間
Table 3 Time required to finish game for each trial.

Ghost object	使用する				使用しない			
同期間隔 (ms)	1000	3000	5000	8000	1000	3000	5000	8000
TeamA	12	106	120	120	166	140	180	180
TeamB	150	147	120	115	157	177	165	165
TeamC	160	156	161	220	154	155	140	168
TeamD	119	115	103	140	94	116	120	140
TeamE	110	113	98	152	108	110	160	189
TeamF	133	122	100	128	98	110	116	120
TeamG	113	122	82	110	106	120	126	150
TeamH	129	110	106	190	110	128	209	189
TeamI	112	100	94	120	101	120	160	160
TeamJ	148	132	125	189	113	213	204	220
平均	123.4	113.4	101.1	147.0	104.3	131.0	156.4	171.0

5,000 ミリ秒までは同期間隔が長くなるにつれて評価が良くなっている．また，表 3 を見ると，ゲームにかかる時間に関しても，完了までにかかる時間は 5,000 ミリ秒まで同期間隔が長くなるにつれ短縮されている．この原因としては，同期間隔が長くなるにつれてゴーストオブジェクトとオリジナルオブジェクトの間隔が適度に開くことによって合成作業がしやすかったことと，同期間隔が長くなるにつれてサーバへのアクセス集中によるボトルネックが解消されたため作業がしやすくなったことがあげられる．これらの結果から仮想共同作業環境における共同作業の際，ユーザにとって共有ゴーストオブジェクトが有効に機能していたこと

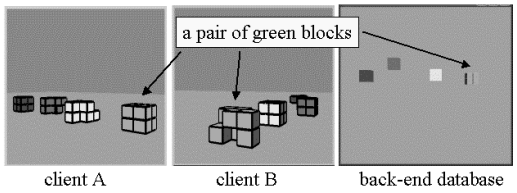

図 12 ゲーム終了時のクライアント A，B およびサーバの状態（共有ゴーストオブジェクトを用いない場合）
Fig. 12 State of virtual world on clientA, clientB and server when the game was finished (without using neighboring ghost objects).

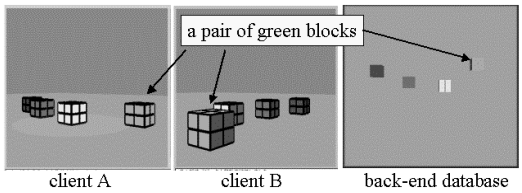

図 13 ゲーム終了時のクライアント A，B およびサーバでの状態（共有ゴーストオブジェクトを用いた場合）
Fig. 13 State of virtual world on clientA, clientB and back-end database when the game was finished (using neighboring ghost objects).

が分かった．この結果に加えて，我々はゴーストオブジェクトを用いることによって 4.1 節（図 7）に示されるような現象が解決されることを確認することができた．**図 12** はゴーストオブジェクトを使わない場合にゲームが終了した時点での，クライアント A，B およびバックエンドデータベースにおける仮想世界の様子である．今回の実験アプリケーションでは，オブジェクトの合成をバックエンドデータベースが検出した時点で，オブジェクトの移動を終了させている．図 12 から分かるように，クライアント A およびバックエンドデータベースではブロックの合成が行われているのにクライアント B では合成が行われないまま移動を終了しており，4.1 節で述べた現象が起きている．一方，ゴーストオブジェクトを用いた場合は**図 13** に示すようにクライアント A，B およびバックエンドデータベースでの仮想世界の様子が一致しており，ゴーストオブジェクトの導入によって 4.1 節であげられた問題点を解決していることが分かる．

考察 2：適切なクライアント–サーバ間の同期間隔

考察 1 により，共有ゴーストオブジェクトを利用すれば，仮想共同作業環境での共同作業の際，ユーザは共有ゴーストオブジェクトを視認することでクライアント–サーバ間の同期間隔の長さに依存せずに作業を行うことができることが分かった．そこで，このアプリケーションにおけるクライアント–サーバの適切な同期間隔について考察する．クライアント–サーバの同期間隔が長くなるにつれてシステム障害時のロールバックが問題となる．たとえばクライアント–サーバ間の同期を 5,000 ミリ秒とした場合，ゴーストオブジェクトは非永続的なものであることからトランザクションがアボートされる際アボート直前のゴーストオブジェクトの位置は保存されず，最悪の場合 5,000 ミリ秒前の状態に戻ることになってしまう．そのためアボート時のゴーストオブジェクトの消失によってユーザの作業に困難が生じない程度の同期間隔をとる必要がある．この場合適切な同期間隔の長さはアプリケーションやユーザが頻繁に行う操作によって大きく異なると考えられ，データベース設計者または利用者が自由にできるように設定できるようにする必要がある．ここでは評価用アプリケーションにおける適切な同期間隔を求める．本アプリケーションではロールバックが起きた場合でもユーザが違和感なく操作を再開できる距離を 2.0 メートルとし，サブトランザクションが発行されてから次のサブトランザクションが発行されるまでのオブジェクトの移動距離が 2.0 メートル以内である割合がゲームが終了するまでに発行されるサブトランザクションの 2/3 以上であった場合その同期間隔は適切であるとする．**表 4** は Team I（ユーザ q とユーザ r）のゴーストオブジェクトを利用した場合の試行で採取した，移動操作トランザクションのサブトランザクション間の移動距離平均と，移動距離が 2.0 メートル以内である割合である．この場合 5,000 ミリ秒の場合はユーザ q，ユーザ r ともに移動距離が 2.0 メートル以内である割合が 2/3 を割っている．また平均でもユーザ q は移動距離平均が 2.0 メートルを超えている．これらの結果より，本アプリケーションでの適切な同期間隔を 3,000 ミリ秒とした．なお考察 1 で表 2 と表 3 を見た結果ではゲーム終了までの作業時間が最も短く評価が良かったのは 5,000 ミリ秒であったが，クライアント–サーバ間の発行間隔として最も考慮すべきことはシステム障害時のロールバックに対応

表 4 Team I の試行におけるサブトランザクションのオブジェクト移動距離平均とそれが 2.0 メートルを超えている割合
Table 4 Average of distance which user q and user r move object between two sub-transactions, and parcentage when the distance exceeds 2.0 m.

同期間隔 (ms)		1,000	3,000	5,000	8,000
user q	移動距離平均 (m)	0.48	1.75	2.12	5.4
	2m 以内の割合 (%)	96.9	73.7	49.9	39.6
user r	移動距離平均 (m)	0.36	1.09	1.16	5.2
	2m 以内の割合 (%)	100.0	79.4	64.4	36.8

できる同期間隔であると考え，3,000 ミリ秒を最適な同期間隔とした．

考察 3：データベースアクセス量の軽減率

考察 1 で共有ゴーストオブジェクトを用いることによって同期間隔の短さに関係なく高い同期性を保つことができることが示され，考察 2 によってこの実験での適切な同期間隔は 3,000 ミリ秒であることが分かった．共有ゴーストオブジェクト間の通信間隔が 200 ミリ秒であることから，共有ゴーストオブジェクトを用い 3,000 ミリ秒の同期間隔に設定した場合の同期性の高さは，共有ゴーストオブジェクトを用いずに 200 ミリ秒の同期間隔に設定した場合の同期性とほぼ同じであると考えることができる．この考えをもとに，以下の 2 つの場合におけるサーバへのアクセス数を比較した：(1) 共有ゴーストオブジェクトを用い，同期間隔を 3,000 ミリ秒に設定した場合，(2) 共有ゴーストオブジェクトを用いず，同期間隔を 200 ミリ秒に設定した場合．(1) の場合のサーバへのアクセスを求めるために，先の実験における Team I による実験の際にアクセス数を採取したところ 120 秒中 98 アクセスであった．一方 (2) の場合のサーバアクセスは同期間隔ごとにクライアントから送信されるため，アクセス数を単純計算で求めることができる．つまり 1 秒間に 2 ユーザから送信されるアクセス数は 10 であることから，120 秒中のアクセス数は 1,200 となる．これらのアクセス数の比較すると (1) の場合のアクセス数は (2) の場合に比べて約 92%減少しており，共有ゴーストオブジェクトを利用することによって，同期性を低下させることなくアクセス数を大幅に軽減できることが分かった．

今回の評価の結果は実験で用いたアプリケーションの性質およびデータベースサーバの処理能力に依存する結果である．今後，より一般的な評価を行うことで仮想共同作業環境全般においてゴーストオブジェクトが有効であることを示す必要がある．しかし今回用いた評価用アプリケーションでは我々が応用例として考えている共同 3D-CAD で頻繁に行われると思われる移動・合成作業を例に用いており，このアプリケーションにおいてゴーストオブジェクトの有効性が確認できたことは大変意味のある結果であったと考えている．

6. まとめと今後の課題

VWDB2 上に実装した仮想共同作業環境に対して，高い信頼性および同期性を提供するために導入した仮想世界同期法について述べた．共有ゴーストオブジェクトを導入することにより同期間隔が比較的長い場合にも高い同期性を保つようにすることで，データベースサーバへのアクセス数の軽減を図り，サーバの負荷軽減を行った．また，実験によって共有ゴーストオブジェクトの有効性を考察し，実験で用いたアプリケーションでは 92%ものアクセス数を軽減できることが分かった．今後の予定として仮想共同作業環境上で複数人が同じオブジェクトを操作したい場合などの協調作業に対応した同時実行制御の定義を考えている．また，仮想共同作業環境による実用的なアプリケーションを実装する予定である．

参 考 文 献

1) Calvin, J., Dickens, A., Gaines, B., Metzger, P., Miller, D. and Owen D.: The Simnet Virtual World Architecture, *Proc. IEEE VRAIS*, pp.450–455 (1993).
2) Funkhouser, T.A.: Network Topologies for Scalable Multi-User Virtual Environment, *Proc. IEEE Virtual Reality Annual Internal Symposium* (*VRAIS'96*), pp.222–228 (1996).
3) Frecon, E. and Stenius, M.: DIVE: A Scalable network architecture for distributed virtual environments, *Distributed Systems Engineering Journal*, Vol.5, No.3, pp.91–100 (1998).
4) Gausemeier, J., Krumm, H., Molt, T. Gbbesmeyer, P. and Gehrmann, P.: A Database Driven Server For An Internet Based Plant Layout Presentation, *Proc. VRML 2000*, pp.17–22 (2000).
5) Gisi, M.A. and Sacchi, C.: Co-CAD: A Collaborative Mechanical CAD System, *PRESENCE*, Vol.3, No.4, pp.341–350, MIT Press (1994).
6) Gray, J.: The Transaction Concept: Virtues and Limitations, *Proc. International Conference on Very Large Data Bases*, pp.144–154 (1981).
7) HORB ホームページ．http://www.horb.org
8) Kamiura, M., Oiso, H., Tajima, K. and Tanaka, K.: Spatial Viewsand LOD-based Access ControliNVRML-object Database, *Worldwide Computing and Its Application*, pp.210–225 (1997).
9) Masunaga, Y. and Watanabe, C.: The Virtual World Database System — Its Concept, Design and Prototyping, *Advances in Multimedia and Databases for the New Century — A Swiss/Japanese Perspective*, pp.61–70, World Scientific (2000).
10) Masunaga, Y. and Watanabe, C.: Design and Implementation of a Multi-modal User Interface of the Virtual World Database System

(VWDB), *Proc. 7th International Conference on Database Systems for Advanced Application* (*DASFAA'01*) (2001).
11) Meehan, M.: Survey of Multi-User Distributed Virtual Environment, *Course Notes: Developing Shared Virtual Environments*, ACM Press (1999).
12) Moss, J.E.B.: Nested Transactions and Reliable Distributed Computing, *Proc. Symposium on reliability in Distributed Software and Database Systems* (1982).
13) Rusinkiewicz, M. and Sheth, A.: Specification and Execution of Transactional Workflows, *Modern Database Systems*, Kim, W. (eds.), pp.592–620 (1995).
14) Schnabel, M.A. and Kvan, T.: Implementing The First Virtual Environment Design Studio: Architectural Education for the Asian Century, *Proc. 1st ACAE Conference on Architectural Education*, pp.157–166 (2001).
15) Singhal, S. and Zyda, M.: *Networked Virtual Environments: design and implementation*, p.331, ACM Press (1999).
16) Takemura, H., Kishino, F.: Cooperating Work Environment Using Virtual Workspace, *Proc. Conference on Computer-Supported Cooperative Work*, pp.226–232 (1992).
17) 渡辺知恵美，大杉あゆみ，佐藤こず恵，増永良文：仮想世界データベースシステムにおける VWDB スキーマ・ドメイン定義言語の設計と実装，電子情報通信学会第 12 回データ工学ワークショップ 1B-3 (2001).
18) Weikum, G.: Principles and Realization Strategies of Multilevel Transaction Management, *ACM Trans. Database Syst.*, Vol.16, No.1 (1991).

(平成 14 年 10 月 7 日受付)
(平成 15 年 1 月 15 日採録)

(担当編集委員 清木 康，市川 哲彦，佐藤 聡，原 隆浩，細川 宜秀)

渡辺知恵美（正会員）

昭和 50 年生．平成 12 年お茶の水女子大学大学院人間文化研究科複合領域科学専攻修了，理学博士．現在，奈良女子大学大学院人間文化研究科複合現象科学専攻助手．VR システムにおけるデータベース支援について，特に，トランザクション管理，協調作業支援，マルチモーダルインタラクションに興味を持つ．日本バーチャルリアリティ学会，日本データベース学会各会員．

増永 良文（正会員）

昭和 45 年東北大学大学院工学研究科博士課程電気及通信工学専攻修了，工学博士．東北大学電気通信研究所助手，IIASA 研究員，IBM San Jose 研究所客員研究員，図書館情報大学助教授，同教授を経て，平成 11 年 2 月よりお茶の水女子大学理学部情報科学科教授，現在に至る．その間，情報処理学会データベースシステム研究会主査，ACM SIGMOD 日本支部長，情報処理学会監事等を歴任．情報処理学会フェロー．現在，日本データベース学会副会長．専門はデータベース．著書に「リレーショナルデータベース入門」（サイエンス社),「リレーショナルデータベースの基礎—データモデル編」（オーム社）等．電子情報通信学会，日本バーチャルリアリティ学会，日本データベース学会，ACM，IEEE-CS 各会員．