平成 12 年度

学士学位論文

サイバースペース方式の仮想空間構築法の研究

A virtual space building method of cyberspace systems

1010385　　川崎 道雄

指導教員　　島村 和典 教授

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

サイバースペース方式の
仮想空間構築法の研究

川崎 道雄

本研究では、人々の積極的な使用を喚起しより多くの普及を見込める仮想空間通信システムの実現を目指して，空間融合を伴い，かつカスタマイズ性に富む仮想空間ワールドリンク方式を提案した．このリンク方式の特長は次の通りである．

1. リンク先ワールドを空間的に接続可能とすることで，リンク元ワールドから目に見える，移動の際の操作負担を少なく出来る事から，ユーザのワールド間移動を促進することが期待できる．
2. 空間接続可能であるにも係わらず，空間的制約条件を緩く設定できるようにワールド間に一方向リンクを許容している．このことにより，ユーザ自身によるワールドリンク接続のカスタマイズ性が高められる．

このリンク方式について，デモプログラムでの一部機能実現と検証を報告するとともに，考察をまとめている．その要点は以下の 3 点である．

1. ユーザが自分のワールドに新たにワールドリンクする時，ワールドリンクは自由に設定できる．また，自分のワールドを拡張することで設定可能なワールドリンク数を増加できる．
2. あるユーザのワールドに他のユーザからワールドリンクが張られる時，一方向のワール

ドリンクは受ける数に制限は無い.

3. 一方向性のワールドリンクを採用することでループ現象が起こる．ループ現象によってワールド間コミュニケーションに矛盾をもたらす.

今後は，通信機能を加えたプログラム開発を行い．さらに今回提案した方式の有効性の検証を拡げていくことが課題である.

キーワード　仮想空間　リンク方式　カスタマイズ性　空間融合　コミュニケーション

Abstract

A virtual space building method of cyberspace systems

Michio KAWASAKI

A spatial linkage method among virtual spaces is proposed in this article. 'Spatial linkages' mean spatial connections among individual worlds in virtual spaces. The spatial linkage method has two excellent features very useful to increase virtual reality telecommunication system users.

One is a merit that makes people's mobility among worlds higher and thus makes communication among people in the virtual spaces more frequent and active, because it is very easy for people to see and know situations in neighbor worlds without entering them. Another feature is to promote people's customizability concerning connections among worlds with allowing uni-direction spatial linkages.

The proposed method is partially implemented, and the features and the problems of the method are discussed.

First, when a user newly links a world to his own world, a link can be set up without permission. Moreover, the number of links which can be set up by extending one's world can be increased. Next, a link is stretched from others in its world, there is no restriction in the number which receives a one-way link. Finally, a loop phenomenon happens by adopting a one-way link. Inconsistency is brought to the communication between worlds according to a loop phenomenon.

From now on, program development which added the communication function should be performed. And it is a subject to verify the validity of the contents considered this time.

key words Virtual space Link system Customize nature Spatial fusion Communication

目次

図目次

表目次

第 1 章

研究の目的

本研究は，サイバースペースと呼ばれる仮想空間共有通信方式の有効なワールド（ユーザが居る仮想空間）構築を目的にする．具体的な課題は，サイバースペースのサーバ管理を一局集中型から分散型に変えることとワールド間のリンク方式を空間的に融合させることで，ユーザの参加，ワールド構築の容易化，ワールド間移動の促進をねらいとしている．

第 2 章

研究の背景

2.1 技術的背景

WWW（World Wide Web) の幅広い利用により，インターネット上のチャットや，IM（インスタントメッセージ）などリアルタイムコミュニケーションのサービスが確立されている．常時接続サービス，ASDL や CATV などの高速通信網の普及により，よりリアルタイムコミュニケーションが活発になると考えられる．

一方，PC のグラフィックス機能の劇的な発達により，より鮮明な 3 次元グラフィックスのリアルタイム表示を可能とした．さらに，これまではワークステーション等の高価であった 3 次元グラフィック環境が，個人で容易に入手可能とした．

1994 年の VRML の標準化により，仮想空間通信システムの開発が現れ始めた．通信環境の整備と 3D グラフィックス表示性能の向上により，より自然なコミュニケーション，かつ現実社会では体験できないサービスが求められる．現実社会では体験できないサービスとしては，絶滅した生き物を観賞する動物園，古代の町並みを再現する美術館がある．これらのサービスは距離的に離れた人同士が同時に楽むことが可能である．

仮想空間通信システムは，オンラインショッピング，ゲーム，オンライン教育などの分野に応用が期待できる．

2.2 既存の仮想空間通信システム

これまでに数多くのシステム提案（InterSpace[1]，SpaceFusion[2]，DIVE[3]，FreeWalk[4]，Locale[5]）がある．しかし，仮想空間通信システムは一般的に普及していない状況にある．サービス内のワールドが個々に存在する方式の代表例として InterSpace，サービス内のワールドの相対位置を保つ方式の代表例として SpaceFusion を調査し，概説をする．

2.2.1 InterSpace（NTT ヒューマンインタフェース研究所）[1]

実世界で空間的，距離的に離れた人同士が，Internet 上で同じ空間内に一緒にいて，初めての出会いをしたり，あるいは共同して仕事，作業を行なったりする，仮想現実空間を Internet 上に実現させ，発展させることを目的として作られた．システムは，ユーザ参加を管理するサーバ，ワールドを管理するサーバ，音声通信サーバといったように各機能ごとにサーバを設ける機能分散型の構成が選択されている．

ワールド構成は，リンクの移動先を自由にカスタマイズできるので現在の HomePage と似ている．リンクは次のワールド内容を代替オブジェクトとして置き，そのオブジェクトに対してクリック，触れるなどのアクションで次のワールドに画面が切り替わる．代替オブジェクトは次のワールドの情報量に乏しい上に制作の手間がいる．また，ワールド間移動時に起こる，ダウンロード時間のストレスがユーザの積極的なワールド間移動を妨げる原因となる．さらに，ワールドはコンテンツ制作の手間，ダウンロード時間を考慮すると広さに限界が生じる．

2.2.2 SpaceFusion（富士通研究所）[2]

SpaceFusion は現実の都市のコピーを仮想空間通信システム内に作り出し，そこに実際の気象データ，交通状態などのデータをクライアントに選択させて表示させることや，さまざまなマルチメディアコンテンツの配信を目的に作られている．1 クライアントが複数のサーバにアクセスして必要な情報のみを取得し，クライアント側で合成して表示するアーキテクチャによりサーバの負荷を分散させる試みがなされている．

ワールド間の接続は，空間的に繋げられる．また，ワールドの構成は大空間を区割りしてワールドに分られる．サービス内のワールド同士は隣り合わなくとも相対的な位置関係は保たれる．よって，サービス内の自分の居る場所を常に把握することができる．閉鎖感に関しては，空間融合のリンクを採用により問題とはならない．しかし，制作側から見た場合に発展性に問題がある．1 ユーザがある領域を自分のワールドとして作成するときに，ユーザは回りのワールド配置に対して何の干渉を持てない．また，ワールドの広がりは常に外側に限定される．

2.2.3 InterSpace と SpaceFusion の比較

InterSpace と SpaceFusion の利点，欠点をワールドの拡張性，ユーザの閉鎖感，リンク先の情報，自分の位置情報の項目で評価して表 1 にまとめた.

表 2.1 仮想空間システム InterSpace と SpaceFusion の相互比較

	InterSpace	SpaceFusion
ワールドの拡張性	ワールド同士は互いに 同一空間内には無く, 新規追加時において 空間的制約を受けず, ワールドの追加がしやすい.	ワールドの新規追加には 空間的制約がある
ユーザの閉鎖感	広さはユーザのいる ワールドに依存されるため ユーザは閉鎖感を感じる.	ワールドによって構成される 大空間を広げることで 閉鎖感を感じる事は少ない.
リンク先の情報	代替オブジェクトや 文字情報に頼るため 情報量が乏しい.	同一空間内に移動先の ワールドが見えるので 情報が直観的である
自分の位置情報	ワールドは個々が独立して 存在するために, 自分のワールドシステム全体に おける相対位置応報を 習得することが困難.	ワールド間の相互関係が保たれる （共通座標軸を持つ）ため, 大空間における自分の 位置情報を習得可能.

表 1 より従来の仮想空間通信システムでは，例えば InterSpace システムのように，ワールドの発展性を重要視すると移動先ワールドに関する情報が事前に得させくく，ワールド間移動が促進されにくい．また，閉鎖感を感じるという問題がある．一方，SpaceFusion シス

テムのように，サービス全体で大空間を形成する方式は，移動先が見えているためワールド間移動は容易で，移動は促進できる，しかし，新規ワールドの追加に空間的制約がつく問題が課せられる．

ユーザに支持され，発展させ得る仮想空間システムは上記の欠点を克服していることが期待される．

第 3 章

提案する仮想空間システム

3.1 方式の要求条件

2 章で述べた問題を解決するために，本研究は仮想空間システムに以下の機能を実装する．

1. ワールド同士が空間的に接続
2. ワールド間移動においてユーザ側の操作は不要
3. ユーザが自由にカスタマイズ可能

また，従来仮想空間システムでは，サーバ処理の局地集中により，システムがダウンしサービス全体が停止する恐れがある．これまではその問題に対し，機能毎の処理サーバを設けて負荷の集中を避けてきた．しかし，音声処理の多負荷がかかる機能や，認証処理の大人数の処理がかかる機能を処理するサーバでは解決されていない．よって，本研究では peer to peer 方式を採用する．これは各ユーザの端末にサーバ/クライアント機能を持たせることで実現させる．この方式を採用することで，負荷の一局集中を避けるとともに，たとえシステムがダウンしたとしても，全サービスにおける被害の割合は最小限ですむ．

システムの全体構成は，大きく分けてブラウザ部，通信部，サーバ部の 3 部で構成する．

3.2 ワールドリンク方式

本研究では，前述した要求条件「ワールド同士が空間的に接続」より，ワールドリンク方式として外周融合方式と空間切取方式の提案をする．

3.2.1　外周融合方式

2 つのワールドの外周をつなげて空間の融合を持たせる方法．ワールドの広さの制限をユーザに感じさせなくする．InterSpace の欠点で挙げた閉鎖感を解消する利点がある．

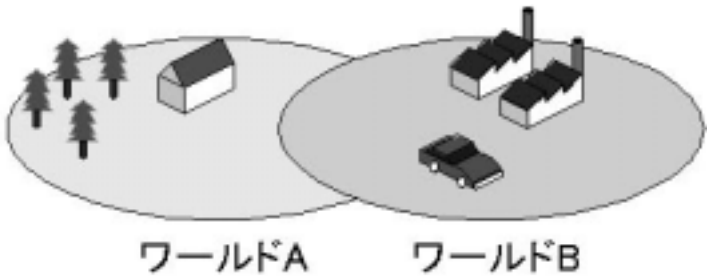

図 3.1　外部融合方式

3.2.2　空間切取方式

他ワールドの一部を自分の所有するワールド内に配置させる方法．他のワールドから切り取ってきて配置した領域にユーザが入ると，その空間が元々存在したワールドに移動をする．さらにワールドリンクでないコピー（ユーザが領域に入っても移動しない）を認める，現在 3D グラフィックスのコンテンツ作成は煩雑でワールドの作成も困難だが，ワールド作成に要する労力を節減できる．後に，述べる方式上の「ワールドリンクの方向性」としては一方向のみとなる．

3.3　ワールドリンクの方向性

機能条件 3 の「ユーザが自由にワールドリンクのカスタマイズ可能」より，ワールドリンクの方向性について双方向のものと一方向のものを設定する．このことにより，サービスにおける相対位置関係を失われ，その結果ユーザは自由にワールドリンクをカスタマイズできる．

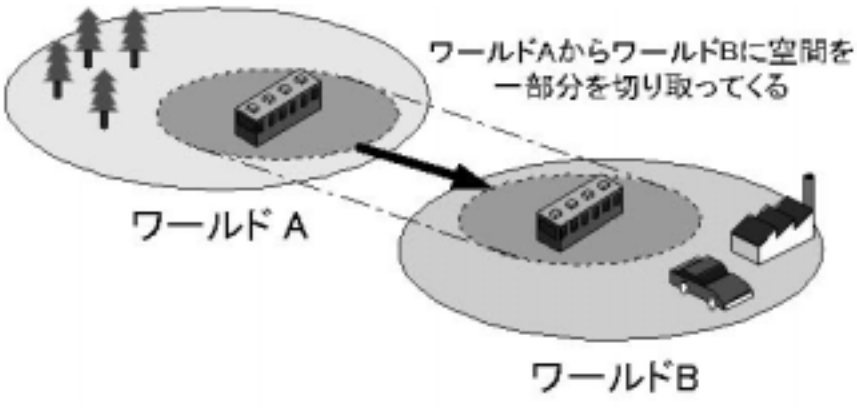

図 3.2　空間切取方式

図 3.3 のように複数のワールドの相互位置関係を X,Y 軸の平面上に表す．ワールド A のある地点を歩きはじめる基点 O と置く．拠点から図 3.3 のように X_1 から Y_1 の移動と Y_1 から X_1 の移動するときに以下の式 3.1 が成り立つ．

$$X_1 + Y_1 \neq Y_1 + X_1 \quad \text{（式 3.1）}$$

これは，ワールド接続に関する空間的制約を弱めたことを示す．

3.4　まとめ

本研究の提案するワールドリンクを持つことで，ユーザは自分のワールドの周りによく行くワールドや人気のあるサイトを，いかにも自分のワールドの拡張であるかように置くことが可能である．また，自分のワールドに接続可能なワールド数に制限がなくなる．

目的場所が明確な移動については，仮想空間を移動するのではなく従来の HomePage でアドレスを直接指定して移動する．もしくは，一度訪問した経歴があるのであれば，HomePage におけるブックマークのような機能により瞬間的に移動することが多いと予想される．つまり，ワールド間の移動という行為は探索が重要な意味を占めると考える．よっ

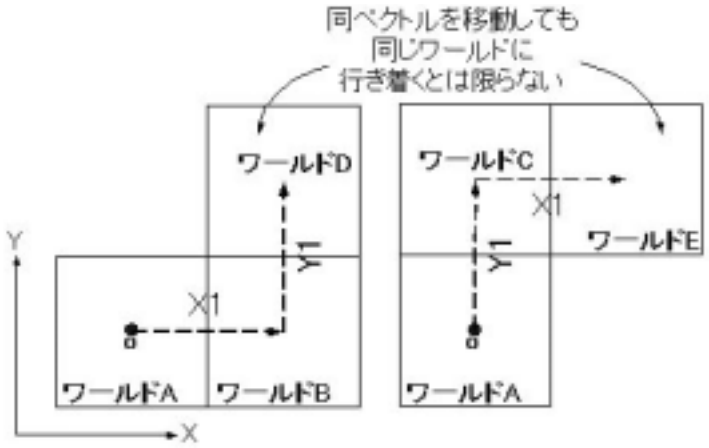

図 3.3　空間的制約の緩和

て，ワールド構成の中で自分の位置を習得するよりもコンテンツを増やすことにユーザの探求行為に対する有利性があることから，ワールドリンクのカスタマイズ性を本研究では重要視する．

第 4 章

実験システム KaleidoSpace

4.1 システム概要

特に実装させる機能は，3.1 方式の要求条件で述べた，今回ブラウザ部，通信部，サーバ部のうち表示部であるブラウザ部に絞り，特にワールドリンクの方向性を 2 種類持つことでユーザのワールドリンク拘束条件の緩和とワールドの階層構造の有利性を図った上で問題点の明確する．

現在自分のいるワールドに付属するワールドリンク記述によって，隣接するワールドが設定され表示できるブラウザを制作する．次のワールドに移動すると，周りのワールド表示は移動後のワールドに付属するワールドリンク記述に従う，そのため周りの情景は一転してる場合がある．移動するワールドによってまわりの情景がさまざまに変わる，これが万華鏡（kaleidoscope）の傾きを変化させることで見えるものが変わることに似ていることから，この仮想空間をカレイドスペース（KaleidoSpace）と呼ぶことにする．

4.2 ワールド構成仕様

これまでのワールドリンク方式の説明において，ワールド同士は重なり合う部分もあることも考慮して述べてきた．しかし，今回の実証する目的はワールドリンクの方向性の拘束条件緩和とワールド構成の階層化の有利性なので，重なり合うことは必要条件ではないとする．重ね合わせが無い条件より空間は直面を合わせる形でワールドリンクを行うことになる．この時，ワールドの形状として適合するものとして立方体を採用する．

立方体であればどの面を合わせたときでも必ず面が一致して合わせられる利点があるからである．また，あるワールドに現在自分がいるのか判断するのに，水平位置座標が正方形であれば自分がどのワールドにいるか領域判定するアルゴリズムを容易にできる利点もある．

その他に，正方形の面が一致して合わせれることにもう 1 つの利点がある．それは双方向ワールドリンクを張るときである．この時，ずれがあると仮定した時，ワールドリンクはそれぞれのワールドから記載されているため，微妙なずれをお互いのワールドリンク記述に正確に書いておかないとワールド移動した後に，直ぐ後ろを向いても元のワールドに戻れない場合がある．これは双方向とは言えない．面がきちんと一致している場合はその時後ろを向いてワールド移動をして戻ることにより必ず元の位置に戻ることが可能である．

ウォークスルーする時，基準の高さとなる面は立方体の底面とし，その座標軸は図 4.1 のようにｘ，ｚ軸は-100～100，y 軸は 0～200 の範囲の値をとるとする．

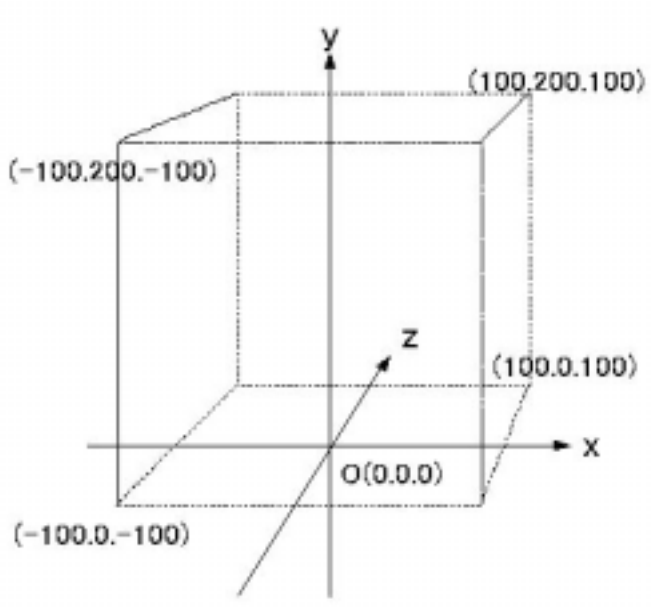

図 4.1　ワールド構成

4.3 実装機能

- ウォークスルー
  - ワールド内およびワールド間をユーザが移動できる
- リンクを空間的につなげる
  - ワールド間移動の視覚化
  - 空間の閉鎖感を緩和する
  - 従来は次のワールド情報を代替オブジェクトや文字情報を制作する必要があったがその手間が無くなる
- ワールドリンクはユーザによってカスタマイズできる
  - 自分のワールドにおける周りの景観を自由に変化させることができる（周りをワールドの延長として見立てて変更できる）
- ワールドリンクは一方向および双方向を規定する
  - 双方向性だけではユーザが自由にワールドリンクを生成することは出来ない
  - 他のワールドから自分のワールドに繋がるワールド数に制限が無くなる（ユーグリッド平面の拘束が無効）
- ワールドリンク記述はワールドデータとは別途用意
  - 将来的にはワールドの構成記述の中に組み込むことが望ましいと考えるが，このプログラムでは管理ファイルを別途規定する
- ワールド表示は現在自分がいるワールドと隣接しているワールドに限定する
  - ワールドリンクの方向性を持たせるため，図 4.2 のように自分のいるワールドから同じ座標軸上であっても異なるワールドが存在する．その時，まったく関連性の無いワールドが重なって表示されることは好ましくない．

4.4 制作環境

- マシン

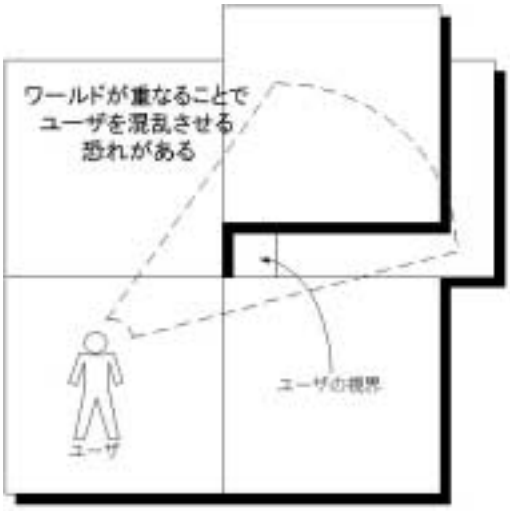

図 4.2 ワールド表示に制限が無い場合

- INTERGRAPH

- CPU
  - Pentium 450Mhz
- メモリ
  - 256M
- グラフィックボード
  - Intense 3D Wildcat 4000
- 開発ツール
  - VisualC++6.0
  - WorldToolKit [*1]
  - 3DStdioMAX [*2]

[*1] SENSE8 社で開発され日本では旭エレクトロニクス社から販売されているバーチャルリアリティ/リアルタイム 3DCG アプリケーションを開発するためのプログラミング環境．リアルタイムシミュレーションを実現するための 1000 以上の高機能ファンクションを持ち，さまざまな VR デバイスに対応している．そのため開発期間やコストを大幅に削減することが可能，およびクロスプラットフォーム（異機種間での互換）の開発環境を実現し，上位機種へのアップグレードや開発環境の共有などが可能．C のライブラリ集．

[*2] discreet 社で開発され日本では Too 社から販売されている 3D グラフィックス作成ツール．映像制作，ゲー

4.5　アルゴリズム

実装機能で述べたように，現在いるワールドからどの方向のワールドに移動したか監視する必要がある．今回の研究では領域が正方形であることから，以下の図 4.3 のような正方形（x 軸：-100～100，z 軸：-100～100）の領域を監視しする．また，隣接するワールドを中心のワールドの位置より，Forward，Backward，Right，Left と置く．

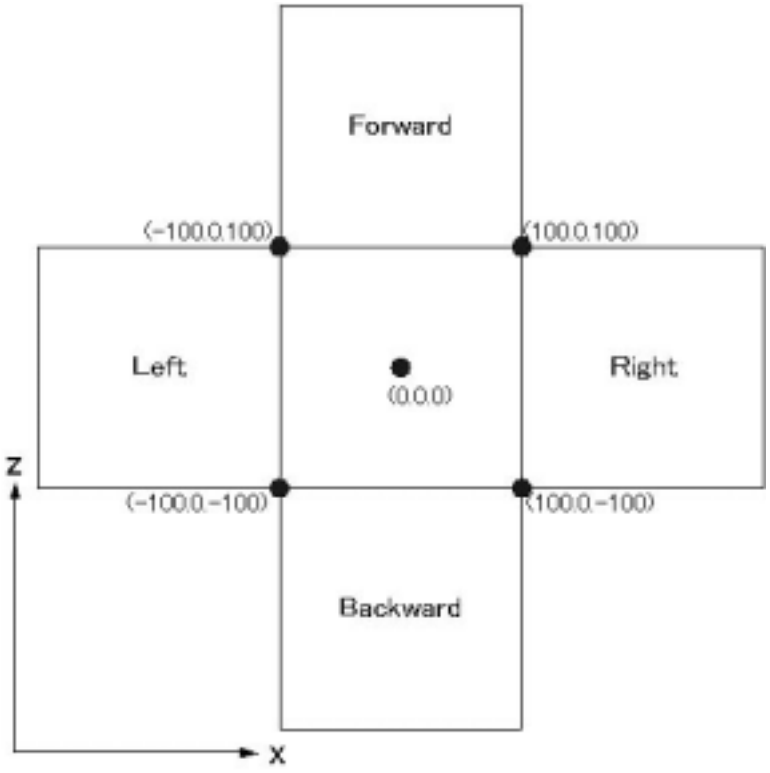

図 4.3　ワールドの位置付け

つまり，300 ＞ z ＞ 100 は Forward，-300 ＜ z ＜-100 は Backward，300 ＞ x ＞ 100 は Right，-300 ＜ x ＜-100 は Left のそれぞれのワールドに移動したことになる．また，前述であげた正六角形の領域を監視する場合，領域の指定が座標軸では取りにくくなる．そのため，変わりの領域判定のアルゴリズムが必要となる．例えば，自分の足元にある地面がどこ

ム制作などでよく利用される．

のワールドに属しているかを常に監視するである．属しているワールドが変わった時にワールド間移動がなされたと判断する．

今回座標軸を利用した空間切り替えのアルゴリズムを使い KaleidoSpace の一部機能を作成した．

4.6　デモンストレーションプログラムの動作

図 4.4 がデモンストレーションプログラムの動作から KaleidoSpace のワールド構成を示したものである．

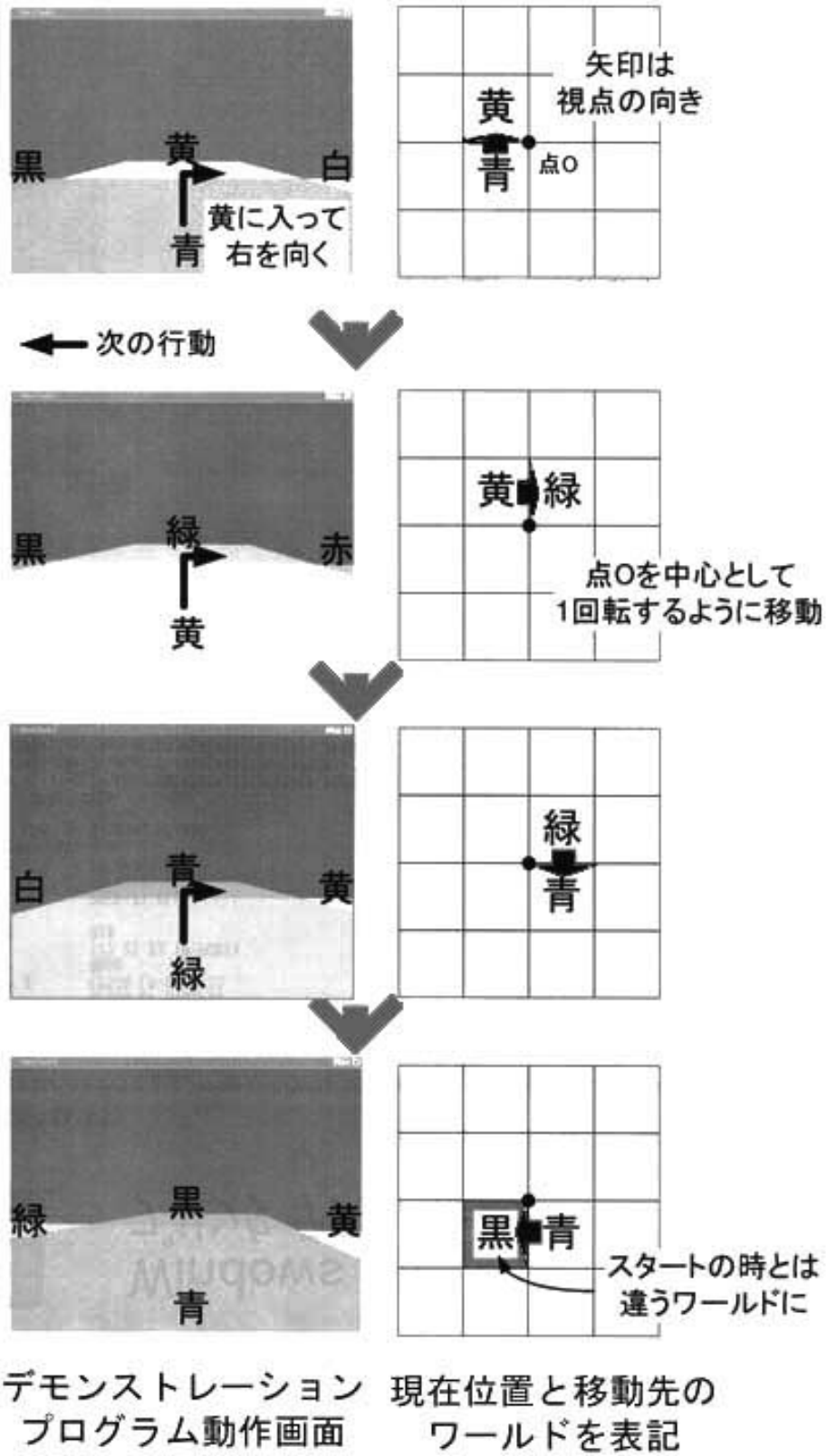

図 4.4　デモンストレーションプログラムの動作

第 5 章

考察

5.1 ワールドリンク制御に対する考察

実験システムでは次の制約条件のもと制作した.

- ワールドの形状は立方体
- 大きさは一定
- ワールドリンクは面を合わせる方式

これらの制約条件のもとでは，ワールドリンク先のワールドを自分のワールドのどの面に合わせるかで，位置が指定できる利点がある．しかし，将来的に制約条件をはずし，ワールドが球の形状，ワールド融合をするという場合，KaleidoSpace のように面でのワールド位置を指定が不可能となる．よって，ワールドの形状，大きさ，融合に制限されない柔軟なワールドリンク記述が求められる.

柔軟なワールドリンク記述の案として，ワールドの中心をベクトルで結ぶ方式を挙げる．このワールドリンク記述の利点としては，ワールドの形状，大きさ，融合に制限されないことの上に，表示する際に処理が少ない.

例えば，提案方式でないワールドリンクするお互いのワールドのリンク地点を記述する場合は，図 5.1 のようになる．このときワールド内の座標系とブラウザの表示座標系の両方を使い表示する必要がある.

一方，提案するワールドリンク記述だと，表示座標軸のみを使うので，表示の際の処理を軽減できる．しかし，ワールドリンクするお互いのワールドが接触する保証のパラメータが

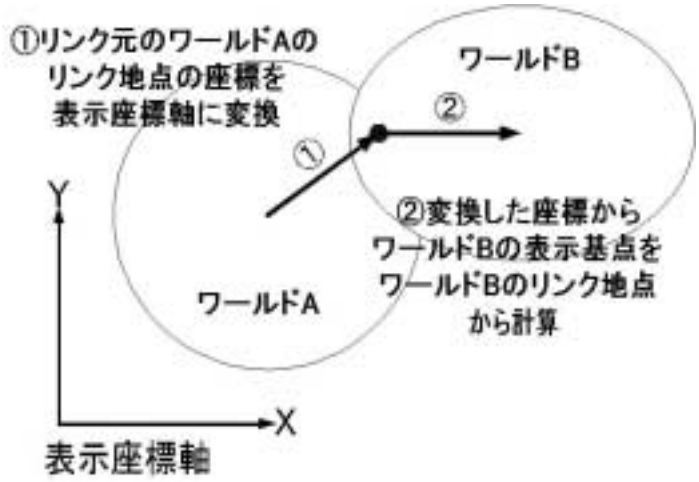

図 5.1 ワールドリンク記述の考察

無い欠点がある．この欠点を補うパラメータを提案する必要がある上に，このパラメータが処理に与える影響を検証する必要がある．

5.2 KaleidoSpace に対する考察

5.2.1 ユーザが他のワールドにワールドリンクする場合

ユーザ α がワールド A を制作している時，ワールド A に属するワールドリンク情報をユーザ α が設定することで，ワールド A に融合するワールドを決定できる．一般的なワールドリンクに，他人のワールドにワールドリンクを張る行為がある．この行為は自分のワールドの空き側面がある限りユーザは自由に設定できる．1 つのワールドには 4 つの側面しかない．しかし，図 5.2 のようにワールド A に新たに自分のワールド A' を接続すれば設定可能なワールドを増やせる．これらの動作を繰り返すことで，ユーザは設定できるワールドリンク数を限りなく増やすことが可能である．また，自分の制作したワールドのみでも拡張していくことが可能である．

さらに，ユーザがワールドリンクを設定可能なのは自分が制作したワールドに限られる．

例えば，他人のワールドのコピー許可があれば，ユーザ A は他人ユーザ C のワールドのコピーを自分のワールドとしておける．コピーワールドのワールドリンクはユーザが自由に設定することができる．図 5.2 の C_A がコピーワールドにあたる．

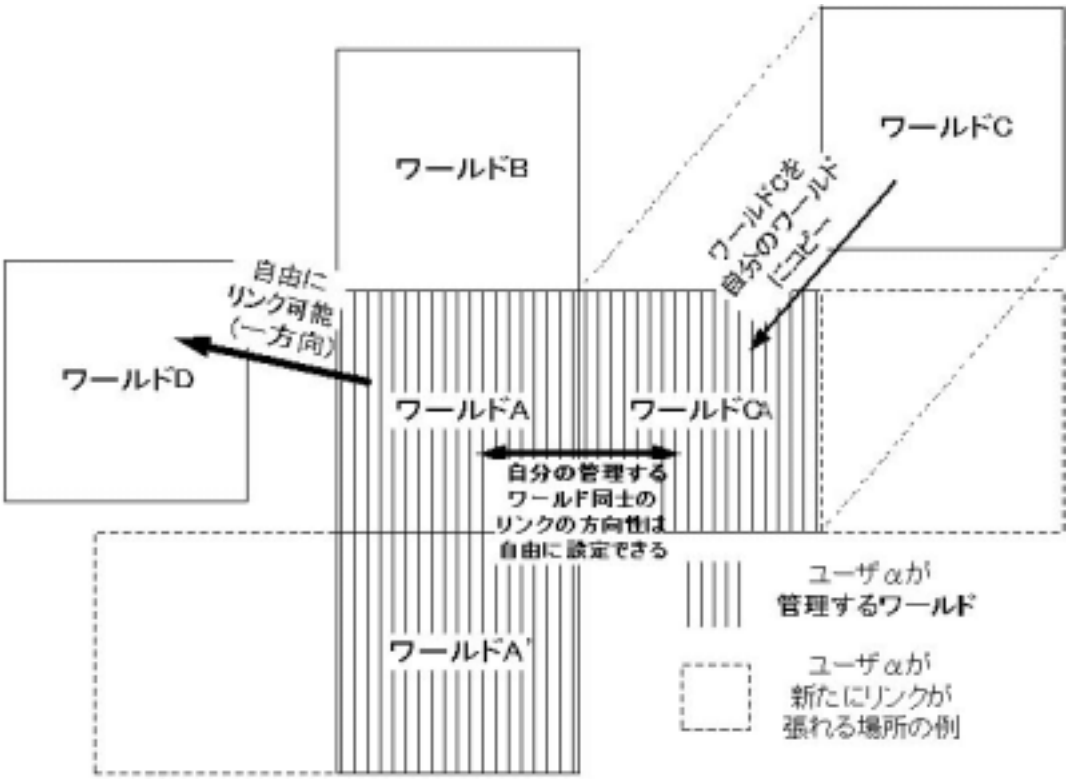

図 5.2　ユーザが自分のワールドに新たにワールドをワールドリンクする場合

以上の考察から，1 つのワールドに設定可能な空間融合ワールドリンクの数は最大４であるが，自分のワールドを接続する，他者のコピーワールドを接続することにより，設定可能な空間融合ワールドリンクの数は増加する．

5.2.2　他人が自分のワールドにワールドリンクする場合

自分のワールド E の側面に空きがある場合，図 5.3 のようにワールド F からワールド E にワールドリンクを張ることは問題ない．はじめはワールド F からの 1 方向ワールドリンクのみであるが，後にワールド E からの記述によっては双方向のワールドリンクとできる．

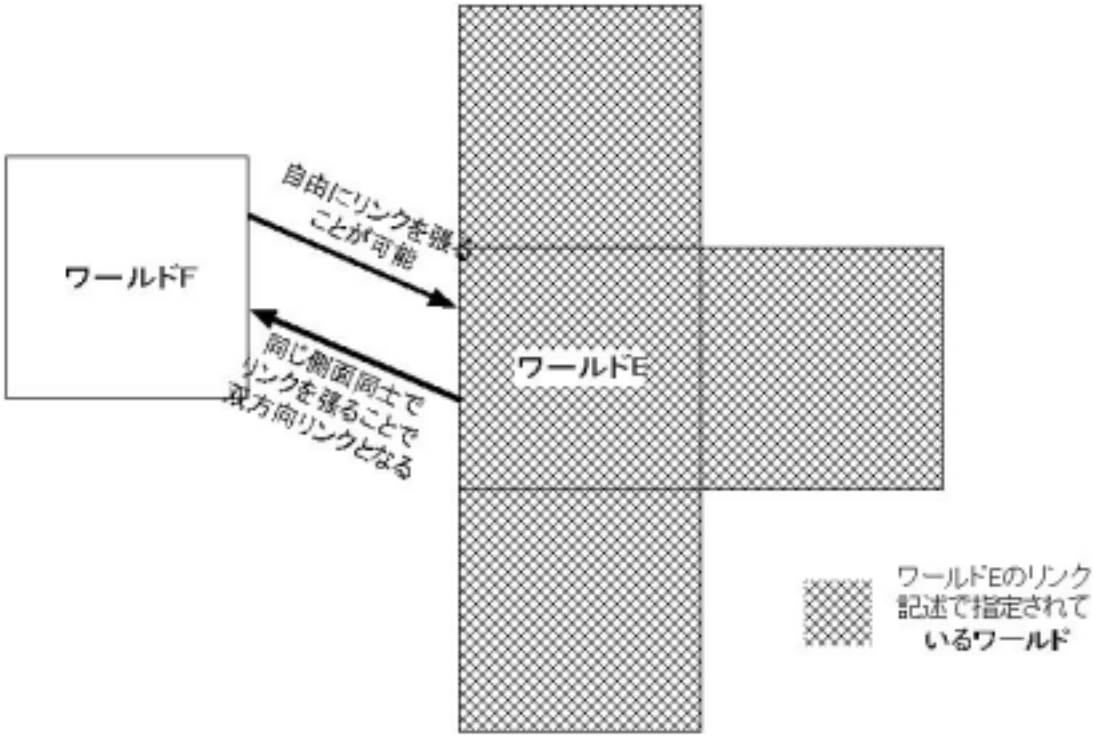

図 5.3　自分の側面に空きがある場合

ワールドの側面がすべてワールドリンク済みの場合は，図 5.4 のように側面がすべてワールドリンクが張られているワールド E に，ワールド F からワールドリンクを張ろうとした場合，1 方向のワールドリンクであれば，ワールドリンクを張ろうとする相手側のワールドの側面がすでに埋められているか否かによらずワールドリンクを張れる．したがって，相手のワールドから自分のワールドに張るワールドリンクの数の制限はない．

以上，自分のワールドに他人からワールドリンクが張られる場合，その数に制限は無い．

5.2.3　ワールド間コミュニケーション

1 方向性のワールドリンクを設けることで，2 つのワールド間コミュニケーションに対して視覚的な矛盾が起こりうる．例えば，ワールドリンクは一方向のみで，ワールド A のワールドリンク記述とワールド B のワールドリンク記述によるワールド構成がそれぞれ図 5.5

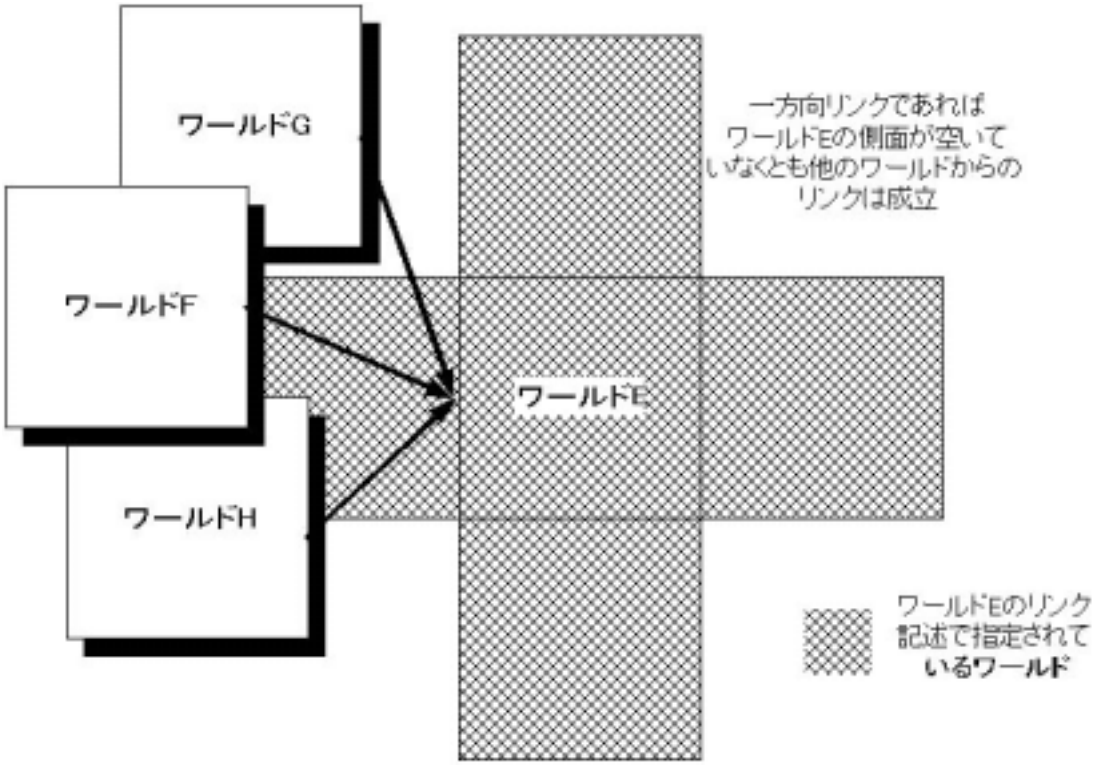

図 5.4　自分のワールドの側面が全てワールドリンク済みの場合

の場合を考える．

この時，ワールド A と B だけに注目すると図 5.6 のようなループ現象が起こる．

ワールド A にユーザ α，ワールド B にユーザ β がいるとする．この時，ユーザ α がユーザ β を追いかけて移動すると，ユーザ β からは α が遠ざかるように見える．その後，ユーザ β の視界から α が消えた瞬間に，ユーザ β の後ろからユーザ α が現れる．また，お互いが近づく時には，終わりの無い追い合いになる可能性がある．これらはコミュニケーション上，ユーザに不要な混乱を与える可能性がある．

これらの問題は，ワールド内の可視範囲が広すぎるために起こりやすくなる．解決策としては，ワールドの広さを広げて，かつ遠くのものを見え難くするか，ユーザの視界範囲を狭めることが考えられる．図 5.8 のようにワールド内のユーザ視界に制限を設けることで，矛盾ある状況を生起し難くする．

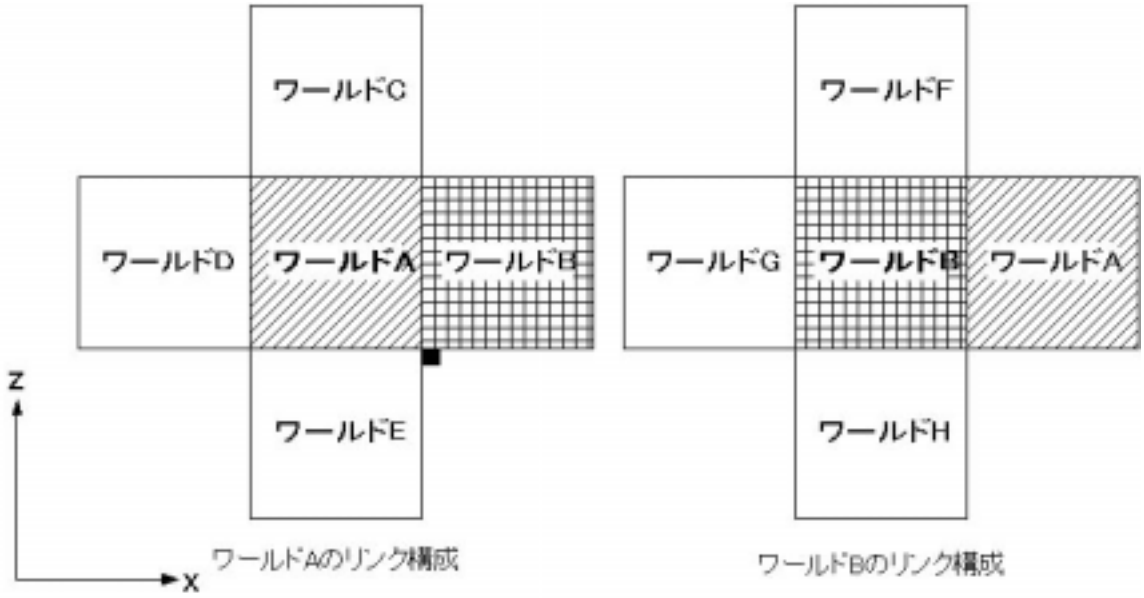

図 5.5　ワールド A とワールド B 構成

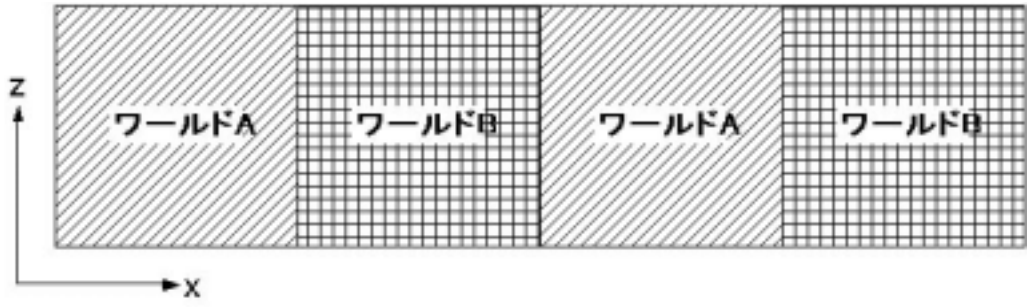

図 5.6　ループ現象

しかし，ワールドを広くしすぎるとユーザのコンテンツ作成にかかる労力が増加することとなり，本研究の目的とは矛盾が生じてくる．また，ワールドが小さすぎてもユーザは視界が狭い空間を移動することになり，ストレスを感じることとなる．よって，ワールド 1 つあたりの広さについて検証する必要がある．

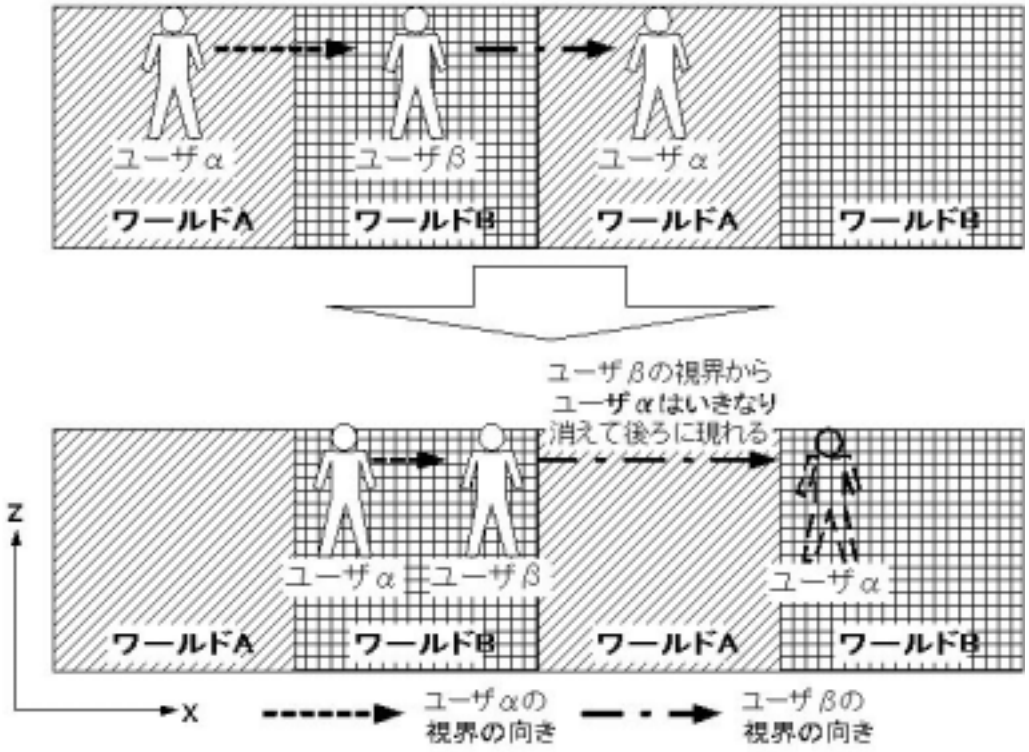

図 5.7 ループ現象上で起こるコミュニケーション問題

5.3 むすび

本研究ではユーザ主導によるワールド発展を見込むためにカスタマイズ性に富んだ空間融合リンク方式を提案した．ワールドリンクの方向性を持つことで，空間的制約を受けずにワールドを拡大しつづけることを可能であることを述べた．また，ワールドリンク制御の考察で，ワールドの中心点を結ぶベクトルによるワールドリンク記述の提案をした．しかし，ワールド同士が接触する保証がない問題点を解決する必要がある．

そして，提案するワールドリンク方式の一部機能を実験システム KaleidoSpace で実装し，そして次の考察を行った．

1. ユーザが自分のワールドに新たにワールドリンクするとき、ワールドリンクは自由に設定できる。また，自分のワールドを拡張することで設定可能なワールドリンク数を増加

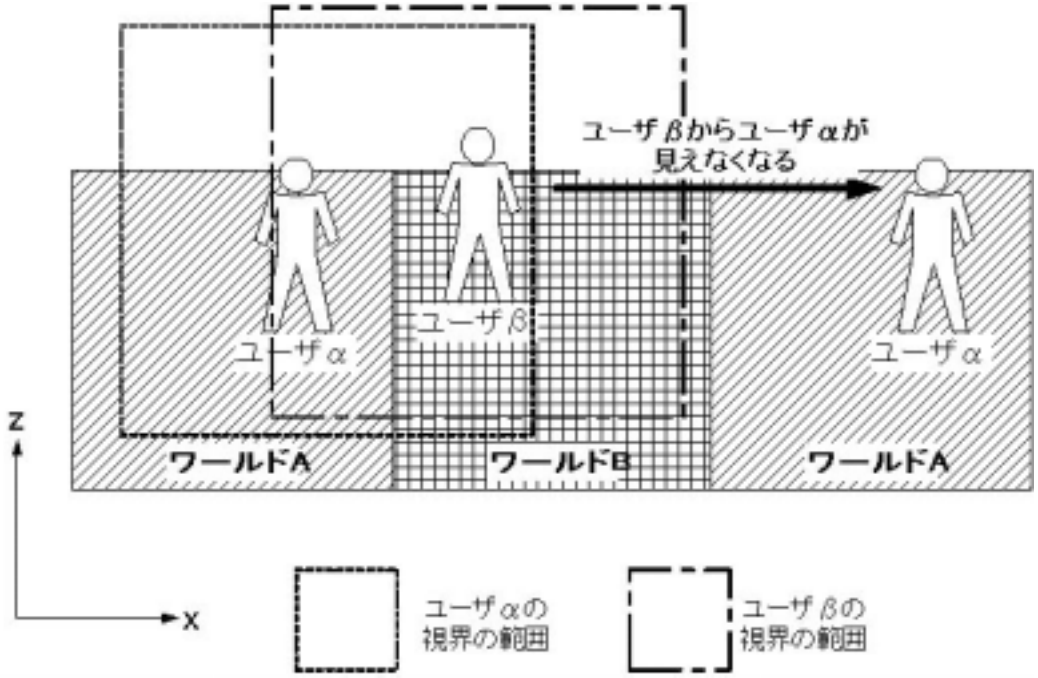

図 5.8　視界制限を設けた場合

できる.

2. 自分のワールドに他人からワールドリンクが張られるとき、一方向のワールドリンクは受ける数に制限は無い.
3. 1 方向性のワールドリンクを採用することでループ現象が起こる。ループ現象によってワールド間コミュニケーションに矛盾をもたらす.

今後は通信機能を加えたプログラム開発を行い．さらに今回考察した内容の有効性を検証していくことが今後の課題である.

謝辞

本研究を行なうにあたり，多くの御教示，ご指導を賜りました島村和典教授に深く感謝します．ならびに, 在学中多くの御助言を頂きました岡田守教授，福本昌弘教授に感謝の意を表します．また，常日頃より多くの御意見，御討論を頂きました通信・放送機構高知トラヒックリサーチセンターの加藤寛治研究員に心より感謝します.

そして，貴重な助言を頂きました中平拓司院生，小林寛征君，橋本江里子さん，野中建史君をはじめとする島村研究室の皆様，通信・放送機構高知トラヒックリサーチセンターの皆様に感謝します.

参考文献

[1] 清末悌之，湯田佳文，山名岳志，加藤洋一，正木茂樹，一之瀬進,「クライアントの性能とサービスの多様性に対応した３次元サイバースペースシステムの機能分散サーバアーキテクチャの提案」，日本バーチャルリアリティ学会論文誌 TVRSJ, Vol.4, No.2, pp.351-356, 1999

[2] Hiroyasu Sugano, Koji Otani, Haruyasu Ueda, Shinichi Hiraiwa, Susumu Endo, Youji Kohda, "SpaceFusion: A Multi-Server Architecture For Shared Virtual Environ-ments," VRML97, February 1997

[3] O. Hagsand, "Interactive MultiUser VEs in the DIVE System," IEEE Multimedia Magazine, Vol 3, No.1, pp.30-39, 1996

[4] H. Nakanishi, C. Yoshida, T. Nishimura and T. Ishida, "FreeWalk: A 3D Virtual Space for Casual Meetings," IEEE MultiMedia, Vol.6, No.2, pp.20-28, 1999

[5] John W. Barrus, Richard C. Waters, David B. Anderson, "Locales: Supporting Large Multiuser Virtual Environments," IEEE Computer Graphics and Applications, 16(6):50-57, November 1996

付録 A

用語解説

サイバースペース

電子技術は，人間の知的活動を代替するだけでなく，知的活動の能力を増幅した．これを可能にしたのは，次に示す 3C という機能である．

－コミュニケーション (Communication)

遠隔地にいる人と人とを結んで，通信を行うことを可能にする働きである．代表例として，電話やパソコン通信が上げられる．

－コンピューティング (Computing)

大脳の理性の働きを人工化したもの．発達した半導体技術や関連技術を導入することによって，応用分野を拡大し続けている．

－コントロール (Control)

制御する機能，調節する機能のこと．コミュニケーション機能やコンピューティング機能などと結び付いて重要な働きをしている．

上記の 3C を結合した技術は，コンピュータコミュニケーションシステムという新しい情報通信環境を作り上げたが，さらに CG，ビデオ，音声などのマルチメディア処理を適用したシステムは，実際には存在しないような仮想的な現実を通信網上に生成する．この仮想的な現実は，バーチャルリアリティと呼ばれる．バーチャル (仮想) とは，「実際には存在しないが，機能や効果において存在しているのと同等である」といった意味で使われる．この仮想的な現実 (仮想現実) によって生成された空間には，多数の人々が集まって対話をしたり情報交換をしたりすることができる．このように，電子技術によって通信網の上に創造された社会空間がサイバースペースである．

3 次元グラフィックス

立体的に描画されたグラフィックスの絵，またはそのためのグラフィックス描画方法．

物体があたかも 3 次元空間に存在するように描画するためには，物体の配置（遠くにある物の一部が，近くにある物によって隠す）や光線の加減（光源に面した部分は明るく，光源とは反対の面は暗くする），物体の材質（鏡面なら光源からの光を反射，透明の物体なら光線を透過させる）などを考慮し，それらによって物体がどのように見えるかを計算しなければならない．このためのグラフィックス描画手法の総称が 3D グラフィックスと呼ばれる．代表的なグラフィックス描画関数としては，Open GL（Windows NT は標準搭載する），QuickDraw 3D などがある．

仮想空間（バーチャルリアリティ）

仮想現実．コンピュータの 3 次元シミュレーションなどにより，人間にとっては現実のイメージに近い仮想的な世界を作ること．典型的な例としては，ディスプレイを搭載したゴーグルを目に装着し，位置や運動量を検出できるセンサーを装備したグローブを手につけ，センサーに応じた 3D の映像をゴーグルに表示することで，コンピュータによって作られた空間をあたかも現実世界のように見せかける装置などが挙げられる．ただしここまで徹底した装置でなくても，コンピュータによる 3 次元グラフィックス，リアルタイムシミュレーションなどを広い意味でバーチャルリアリティと呼ぶ場合も多い．

ワールド

地面，構成物（オブジェクト）などの 3 次元グラフィックスコンテンツおよびマルチメディアコンテンツなどで構成されるユーザ同士のコミュニケーションを提供する空間．

リンク

インターネットであるウェブページ上から別のウェブページへ接続すること．World-

WideWeb においては，あるウェブページが別のウェブページとリンクしている場合，ページ領域内に色文字や下線付きなどの強調文字，色枠付き画像などが表示されている．そしてそれらをクリックすることで別のページ（リンクページ）に接続できる．

サーバースペースではウェブページをワールドと置いて考える．

付録 B

KaleidoSpace Program

```
#include "wt.h"

#define FORWARD 1
#define BACKWARD 2
#define RIGHT 3
#define LEFT 4

static void actionfn(void);
FLAG SetSpaceData(char *name);
int CheckArea(void);
void ChangeSpace(int area);

WTnode  *centerSpace, *forwardSpace, *backwardSpace,
        *rightSpace, *leftSpace;
char    centerName[64], forwardName[64], backwardName[64],
        rightName[64], leftName[64];

void main(int argc, char *argv[])
{
    WTnode *root;
    WTnode *light;
    WTviewpoint *view;
    WTsensor *mouse;
    WTp3 dir, pos;
```

```
/* Initialize the universe.
 * (This must be the 1st WorldToolKit call.)
 */
WTuniverse_new(WTDISPLAY_DEFAULT, WTWINDOW_DEFAULT);

/* get a pointer to the universe root node */
root = WTuniverse_getrootnodes();

/* set up the position of a light source */
light = WTlightnode_newdirected(root);
dir[X] = 0.58f;
dir[Y] = 0.58f;
dir[Z] = -0.58f;
WTlightnode_setdirection(light, dir);

/* Initialize a Mouse sensor */
mouse = WTmouse_new();
if (!mouse) WTerror("Couldn't find mouse\n");

/* Attach the sensor to the viewpoint. */
view = WTwindow_getviewpoint(WTuniverse_getwindows());
WTviewpoint_addsensor(view, mouse);

/* Set the action function so that button presses will be acted on. */
WTuniverse_setactions((void *)actionfn);
WTkeyboard_open();

/* Initial setting of worlds */
SetSpaceData(argv[1]);

/* Initial setting of viewpoint */
```

```
    pos[X] = pos[Z] = 0.0f;
    pos[Y] = -50.0f;
    WTviewpoint_setposition(view, pos);
    WTsensor_setsensitivity(mouse, 0.01f*WTnode_getradius(root));

    /* Prepare the universe for start of the simulation. */
    WTuniverse_ready();

    /* enter the main simulation */
    WTuniverse_go();

    /* all done; clean everything up. (This must be the last WTK call.) */
    WTuniverse_delete();
}

static void actionfn(void)
{
    short key;
    int area;

    area = CheckArea();
    if(area != 0) ChangeSpace(area);

    /* Keyboard operation */
    key = WTkeyboard_getlastkey();

    if(key == WTKEY_ESC)
        WTuniverse_stop();
    if(key == 'i')
        printf("%f\n", WTuniverse_framerate());

    if(key == '1')
```

```
        WTuniverse_setrendering(WTRENDER_WIREFRAME);
    if(key == '2')
        WTuniverse_setrendering(WTRENDER_SHADED);
    if(key == '3')
        WTuniverse_setrendering(WTRENDER_BEST);
}

FLAG SetSpaceData(char *name)
{
    /* World arrangement algorithm */
    FILE *file;
    char dataName[64], vrmlFileName[64], linkFileName[64], temp[64];
    WTnode *root;
    WTp3 pos;

    strcpy(dataName, name);
    strcpy(centerName, name);
    strcpy(vrmlFileName, dataName);
    strcpy(linkFileName, dataName);
    strcat(vrmlFileName, ".wrl");
    strcat(linkFileName, ".li");

    root = WTuniverse_getrootnodes();

    centerSpace = WTmovnode_load(root, vrmlFileName, 1.0f);

    file = fopen(linkFileName, "r");
    if(!file){
        printf("%s が開けません.\n", linkFileName);
        return FALSE;
    }
```

```
fscanf(file, "%s", temp);
strcpy(forwardName, temp);
strcpy(vrmlFileName, temp);
strcat(vrmlFileName, ".wrl");
forwardSpace = WTmovnode_load(root, vrmlFileName, 1.0f);
WTp3_init(pos);
pos[Z] = 200.0f;
WTnode_settranslation(forwardSpace, pos);

fscanf(file, "%s", temp);
strcpy(backwardName, temp);
strcpy(vrmlFileName, temp);
strcat(vrmlFileName, ".wrl");
backwardSpace = WTmovnode_load(root, vrmlFileName, 1.0f);
WTp3_init(pos);
pos[Z] = -200.0f;
WTnode_settranslation(backwardSpace, pos);

fscanf(file, "%s", temp);
strcpy(vrmlFileName, temp);
strcpy(rightName, temp);
strcat(vrmlFileName, ".wrl");
rightSpace = WTmovnode_load(root, vrmlFileName, 1.0f);
WTp3_init(pos);
pos[X] = 200.0f;
WTnode_settranslation(rightSpace, pos);

fscanf(file, "%s", temp);
strcpy(leftName, temp);
strcpy(vrmlFileName, temp);
strcat(vrmlFileName, ".wrl");
leftSpace = WTmovnode_load(root, vrmlFileName, 1.0f);
```

```
    WTp3_init(pos);
    pos[X] = -200.0f;
    WTnode_settranslation(leftSpace, pos);

    fclose(file);

    return TRUE;
}

int CheckArea(void)
{
    /* World judging algorithm */
    WTp3 pos;
    WTviewpoint *view;

    view = WTwindow_getviewpoint(WTuniverse_getwindows());
    WTviewpoint_getposition(view, pos);

    if(100 .0f < pos[Z] || pos[Z] < 300.0f) return FORWARD;
    if(-300 .0f < pos[Z] || pos[Z] < -100.0f) return BACKWARD;
    if(100 .0f < pos[X] || pos[X] < 300.0f) return RIGHT;
    if(-300 .0f < pos[X] || pos[X] < -100.0f) return LEFT;

    return 0;
}

void ChangeSpace(int area)
{
    /* World change algorithm */
    WTnode *root;
    WTviewpoint *view;
    WTp3 pos;
```

```
    root = WTuniverse_getrootnodes();
    WTnode_delete(centerSpace);
    WTnode_delete(forwardSpace);
    WTnode_delete(backwardSpace);
    WTnode_delete(rightSpace);
    WTnode_delete(leftSpace);

    view = WTwindow_getviewpoint(WTuniverse_getwindows());
    WTviewpoint_getposition(view, pos);

    if(area == FORWARD){
        SetSpaceData(forwardName);
        pos[Z] = -100.0f;
        WTviewpoint_setposition(view, pos);
    }
    else if(area == BACKWARD){
        SetSpaceData(backwardName);
        pos[Z] = 100.0f;
        WTviewpoint_setposition(view, pos);
    }
    else if(area == RIGHT){
        SetSpaceData(rightName);
        pos[X] = -100.0f;
        WTviewpoint_setposition(view, pos);
    }
    else if(area == LEFT){
        SetSpaceData(leftName);
        pos[X] = 100.0f;
        WTviewpoint_setposition(view, pos);
    }
}
```