社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－HI－121 (9)
2006／11／22

手書き作図アプリケーションの効率的開発のための仮想デバイス層の提案

川添昌俊† 佐賀聡人†

我々は汎用的手書き図形認識エンジン FSCI を開発し，さらにそれを利用することで複数の手書き作図アプリケーションを見通しよく開発できることを示してきた．これらの手書き作図アプリケーションでは，アプリケーションごとに個別に開発する「アプリケーション層」と FSCI が共通に認識サービスを行う「論理層」の 2 層構造に基づく設計がされており，これにより図形認識エンジンの共用が実現されていた．本稿では，手書き作図アプリケーションを従来よりもさらに見通しよく設計するためにアプリケーション層と論理層の中間に新たに「仮想デバイス層」を設けることを提案する．さらに，実際に仮想デバイス層を含めた 3 層構造に基づいて従来の手書き作図アプリケーションを再構成した結果，手書き作図アプリケーションのコードの共用化が進み，従来よりも効率のよい開発が可能となったことを示す．

Virtual Device Layer for Efficient Development of Sketch-Based CAD Applications

MASATOSHI KAWAZOE† and SATO SAGA†

We have already developed a general-purpose freehand curve recognition engine FSCI. Then, we have showed that different sketch-based CAD applications have been developed by utilizing FSCI as a common recognition engine. In order to realize the shared use of FSCI, each of the sketch-based CAD applications was designed to be composed of 2 layers, one of which is "application layer," and the other of which is "logical layer" where FSCI provides for freehand curve recognition service. This paper proposes to introduce yet another layer, that is "virtual device layer," between the application layer and the logical layer in order to provide more clear designing framework for the sketch-based CAD applications. Furthermore, through reconstructing the sketch-based CAD applications based on the proposed 3-layer design framework, we show that the introduction of virtual device layer contributes to expansion of source-code sharing and enhancement of efficiency in the application development.

1. はじめに

我々は既にスタイラスによって入力された手書き曲線を幾何曲線列として認識するアルゴリズムとしてファジィスプライン曲線同定法 (FSCI) を提案した[1),2)]．また，FSCI を図形認識エンジンとして利用し，これに表示一体型タブレットを用いたユーザインタフェースを組み合わせることで 2 次元手書き作図アプリケーション SKIT を開発した[3)]．一方，同様に FSCI を図形認識エンジンとして利用しながら，これにヘッドマウントディスプレイ (HMD) や 3 次元位置センサなどの仮想現実感システムを用いたユーザインタフェースを組み合わせることで 3 次元手書き作図アプリケーション BlueGrotto を開発した[4),5)]．これら SKIT と BlueGrotto では，入力デバイスやユーザインタフェースの違いをアプリケーション層で吸収しつつ論理層における FSCI の図形認識サービスを共用することで，見通しのよい手書き作図アプリケーションの設計を実現していた．しかし，SKIT と BlueGrotto の開発を通じて，両者のアプリケーション層にはなお本質的に共用できる部分があることが明らかとなってきた．本稿ではこの共用部分を仮想デバイス層としてアプリケーション層から分離することで手書き作図アプリケーションを従来よりもさらに見通しよく開発できることを示す．

以下では，まず 2 節で従来の手書き作図アプリケーションとその問題点について概説する．3 節では 4 節での提案の準備として 2 階層からなる従来の手書き作図アプリケーションの構成について整理する．4 節

† 室蘭工業大学工学部情報工学科
Department of Computer Science and Systems Engineering, Muroran Institute of Technology

では仮想デバイス層を新たに提案することにより，従来のアプリケーション層における共用部分の分離を図る．さらに5節では実際に仮想デバイス層を含めた3層構造に基づいてSKITとBlueGrottoを再構成した結果，これらのソースコードの共用化が進み，従来よりも効率のよい開発が可能となったことを示す．

2. 従来の手書き作図アプリケーションとその問題点

汎用手書き図形認識エンジンFSCIと従来の手書き作図アプリケーションSKITおよびBlueGrottoの概要を示す．また，従来のSKITおよびBlueGrottoのアプリケーション層には開発効率上改善すべき問題点があることを示す．

2.1 汎用手書き図形認識エンジン FSCI

FSCIは文献1)，2)で提案され，文献4)で3次元に拡張されたオンライン手書き図形認識アルゴリズムであり，3次元空間中の一筆書きの手書き曲線を幾何曲線列として認識する．

ここでFSCIは

- 作図で重要な7種類の幾何曲線(線分，円，円弧，楕円，楕円弧，閉じた自由曲線，開いた自由曲線)をすべて認識できる(網羅性)，
- 作図アプリケーションに依存した背景知識を利用することなく手書き描画動作情報のみを元にして認識を行う(独立性)，

という特徴を有しており，このためFSCIを種々の手書き作図アプリケーションの汎用的な手書き図形認識エンジンとして活用することが可能である．

なお，このFSCIと文献6)で提案された逐次型ファジィスプライン曲線生成法(S-FSCG)を組み合わせることで重ね書き修正描画にも対応する幾何曲線認識が実現されており，本稿ではS-FSCGを含めた認識アルゴリズム全体をFSCIと呼ぶこととする．

2.2 手書き作図アプリケーション SKIT

SKITは文献3)で提案された平面描画型の手書き作図アプリケーションであり，入出力デバイスとして**図1**のような表示一体型のタブレットとスタイラスを利用する．これは幾何曲線から構成される2次元作図を手書きで行えるユーザインタフェースを実現しており，FSCIを手書き図形認識エンジンとして利用している．SKITは既存の様々なクライアント作図アプリケーション(CADソフト，3次元CG制作ソフト，ベクタグラフィックスデザインソフトなど)のウィンドウ上に覆い被さるようにして起動し，このクライアント作図アプリケーションに手書き図形入力サービスを提供するフロントエンドプロセッサとして動作する．

2.3 手書き作図アプリケーション BlueGrotto

BlueGrottoは文献4)，5)で提案された空間描画型の手書き作図アプリケーションであり，入出力デバイスとして**図2**のような没入型仮想現実感システムを利用する．出力デバイスとしては，立体視可能な「ステレオHMD」が用いられている．一方，入力デバイ

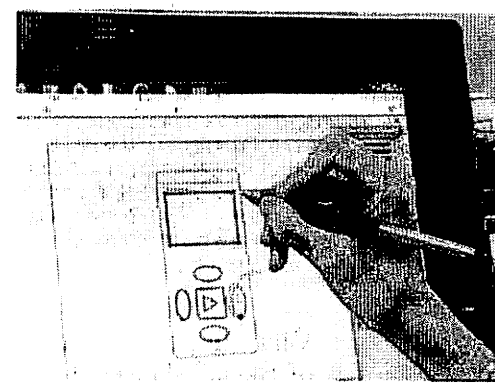

図1 SKITの使用風景

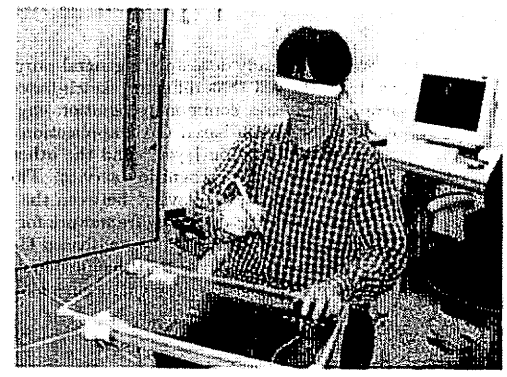

図2 BlueGrottoの使用風景

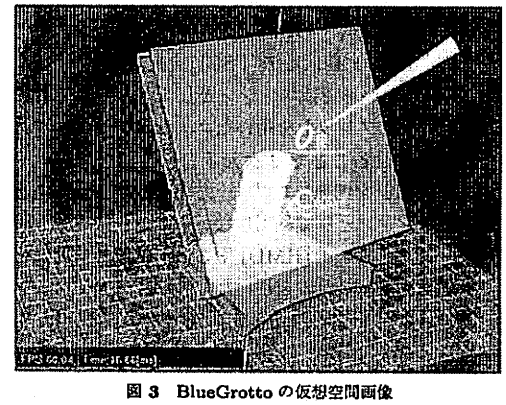

図3 BlueGrottoの仮想空間画像

スとしては，空中描画を行うための「3次元位置センサ搭載スタイラス」，仮想空間中の幾何立体図形の姿勢操作を行うための「3次元コントローラ」，ユーザがかけているステレオHMDの位置を計測し視点位置を更新するための「3次元位置センサ」が用いられている．

BlueGrottoは，図3のように，幾何立体図形から構成される3次元モデリングを仮想空間中の空中手書き動作で直接行えるユーザインタフェースを実現している．ここでは幾何立体図形の手書き入力を複数の幾何曲線列の組を手書きで空中に配置することで実現している*．そのため，BlueGrottoにおいても幾何曲線列を認識するFSCIを手書き図形認識エンジンとして利用している．

2.4 開発効率上の問題点

SKITとBlueGrottoのような入出力デバイスやユーザインタフェースの異なった手書き作図アプリケーションであっても，その違いをアプリケーションで吸収しつつ汎用手書き図形認識エンジンFSCIを共用することで，これらをある程度効率的に見通しよく開発することが可能であった．実際SKITにおけるFSCIのソースコードの割合はステップ数換算で約50%，BlueGrottoのそれも約50%に達した．ただしここで，SKITとBlueGrottoの非共用部分に着目すると，両者の間には離散化データの処理に関してさらにまだ本質的に共用できる部分があることが開発を通じて明らかとなってきた．したがってこの部分を共用することができれば手書き作図アプリケーションの開発効率がさらに向上すると期待される．

しかし，従来の実装ではFSCI以外の共用化について十分な考察が行われておらず，個別の手書き作図アプリケーションの設計段階でこのような共用化を実現することが困難であるという問題があった．

3. 従来の手書き作図アプリケーションの構成

ここでは4節での提案に先立ち，手書き作図アプリケーションが関わる3つの空間の概念を整理した後，2階層からなる従来の手書き作図アプリケーションの構成について整理する．

3.1 空間の概念の整理

手書き作図アプリケーションのおおまかな流れを追うと図4のようになる．まずユーザは現実世界の3次元空間(ここではこれを「物理空間」と呼ぶことにする)でスタイラスを用いて描画を行う．描画はスタイラスデバイスによりサンプリングされ離散的な空間(ここではこれを「仮想デバイス空間」と呼ぶことにする)に点列として取り込まれる．点列は手書き図形

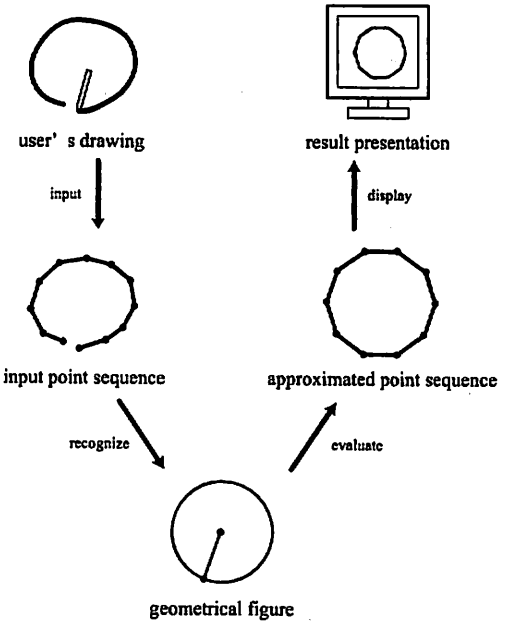

図4 手書き作図アプリケーションの処理の流れ

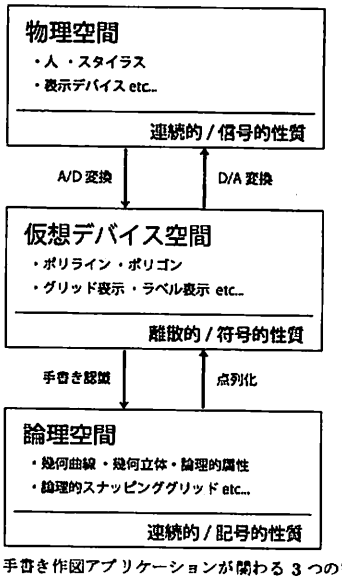

図5 手書き作図アプリケーションが関わる3つの空間

* 具体的には幾何立体図形をスイープもしくはスキニングで表現する．例えば，経路として幾何曲線「線分」を手書き入力し，また断面としてこれに垂直な幾何曲線「円」を手書き入力することで，円を線分でスイープした幾何立体図形「円柱」を手書き入力できる．

認識アルゴリズムにより連続的作図空間 (ここではこれを「論理空間」と呼ぶことにする) 中の幾何曲線として認識される．幾何曲線は点列近似され仮想デバイス空間に配置される．近似点列はディスプレイデバイスによって物理空間に投影される．このことから手書き作図アプリケーションは図 5 のような「物理空間」，「仮想デバイス空間」，「論理空間」の 3 つの空間に関わっていることがわかる．

3.1.1 物理空間

ユーザがスタイラスやその他の入力デバイスを操作する現実の 3 次元空間である．また，処理結果も表示デバイスを通じてこの空間にいるユーザへ光として提示される．本質的に連続的世界であり，そこで観測される操作 (動作) や光は信号的性質を持つ．

3.1.2 仮想デバイス空間

物理空間での操作が入力デバイスによって A/D 変換された結果および論理空間のオブジェクトが点列化された結果を保持する空間である．この空間に保持されるオブジェクトとして，具体的には，点列化描画曲線，点列化幾何曲線，ラベル表示，グリッド表示などがある．ただし，これらのオブジェクトはすべてデバイスの解像度に応じて具現化された表現であり，ポリラインもしくはポリゴンデータとして点列化されていることに注意する．この空間は離散的であり，そこに配置されているオブジェクトは符号的性質を持つ．

3.1.3 論理空間

幾何情報を保持する論理的作図空間である．この空間に保持されるオブジェクトとしては手書き図形認識アルゴリズムにより認識された幾何曲線やそれの組み合わせで表現される幾何立体図形，および論理的スナッピンググリッド，論理的属性などがある．この空間は本質的に連続的であり，そこに配置されるオブジェクトは記号的性質を持つ．

3.2 2 層構造によるアプリケーションの構成

手書き図形認識エンジン FSCI を共用する従来の手書き作図アプリケーション SKIT および BlueGrotto の構成を前述の 3 つの空間と対応付けて整理すると，これらは図 6 に示すように「アプリケーション層」と「論理層」の 2 層で構成されていると考えられる．ただし実線矢印は機能間での処理の依頼元から依頼先への関係を示し，点線矢印は機能から空間へのアクセスを示している．ここでアプリケーションは論理空間にアクセスする手書き図形認識エンジン FSCI を共用しており，これにより手書き図形認識処理を依頼する呼び出し部分は共通化されている．一方それ以外の，論理空間にアクセスして認識結果を取得・加工するといった処理や，仮想デバイス空間にアクセスしてデバイスからの入力データを点列化して登録するといった処理などはすべてアプリケーション側で個別に実装している．

このような設計は自由度の高い実装が可能になる反面，アプリケーション開発者が基本的なオブジェクトコンテナの管理やエンジンの操作などをすべて行う必要があり，開発者にかかる負担が大きい．また，アプリケーション側から仮想デバイス空間と論理空間の両方にアクセスするため，どちらの空間に対する操作を行っているかということを常に意識する必要があり，実装時に混乱を招きやすいという側面もある．

3.3 アプリケーション開発作業の重複

SKIT および BlueGrotto の開発を通じて明らかとなった従来の手書き作図アプリケーションの開発作業の重複部分を整理する．

スタイラスデバイスからの入力データに着目すると，これを時刻情報つき点列 (時系点列) に変換するところまでの処理はデバイスの性能や性質に強く依存しており本質的にアプリケーションごとに個別に開発されるべきものである．これに対し，時系点列を仮想デバイス空間に展開した後の処理はデバイスに依存しておらず，この処理は基本的にすべてのアプリケーション

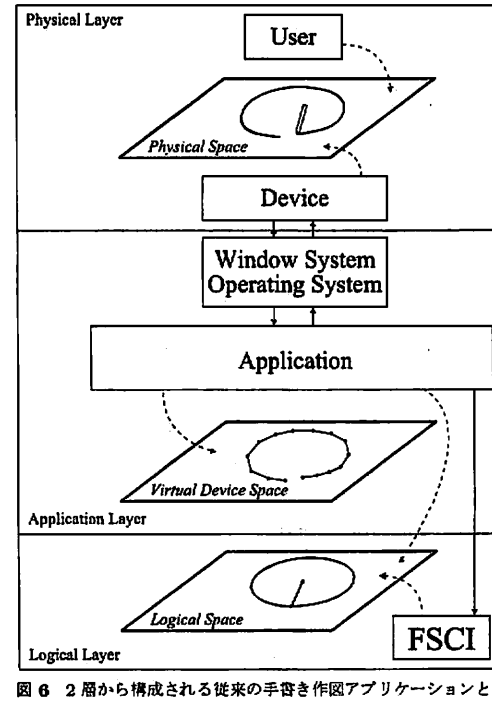

図 6 2 層から構成される従来の手書き作図アプリケーションと 3 つの空間との関係図

で共用化できうるものである．しかし従来の2層構造のアプリケーションではこの処理が分離されておらず，同様の処理がアプリケーション間で重複して開発されていた．

一方，モデリング機能に着目すると，SKITでは幾何曲線列を最終認識結果とするのに対し，BlueGrottoでは幾何立体図形を最終認識結果としており，最終認識結果を生成するためのユーザインタフェースや処理は基本的にアプリケーションごとに個別に開発せざるをえない．しかし，これらの最終認識結果はいずれも論理空間における幾何曲線列の認識処理に立脚しており，この処理には異なるアプリケーション間で共用化できうる処理が多く含まれる．ところが，従来の2層構造のアプリケーションではこのような共用化部分を容易に分離できる構造になっておらず，ここでもやはり同様の処理がアプリケーション間で重複して開発されていた．

以上をまとめると，従来の手書き作図アプリケーションの開発作業における無駄な重複作業は，仮想デバイス空間や論理空間に直接アクセスする部分の処理の共用化が可能にもかかわらず，これらの処理が個々のアプリケーションに委ねられていたことによって発生していたと考えられる．

4. 手書き作図アプリケーションへの仮想デバイス層の導入

ここでは従来のアプリケーション層から共用部分を分離し，手書き作図アプリケーションの開発効率をさらに向上をさせるために，MVCソフトウェアパターンで構成された仮想デバイス層を新たに導入することを提案する．

4.1 仮想デバイス層によるアプリケーションと空間の分離

3.3での検討に基づき，従来の手書き作図アプリケーションのアプリケーション層から仮想デバイス空間および論理空間へ直接アクセスする部分を「仮想デバイス層」として抽出分離することで，図7のように手書き作図アプリケーションを「アプリケーション層」，「仮想デバイス層」，「論理層」の3層構造で再構成することを新たに提案する．この構成によりアプリケーション層が仮想デバイス空間および論理空間のオブジェクトに直接アクセスすることがなくなり，アプリケーション層は物理層のデバイスの機能や仮想デバイス層の機能とのみインタフェースをもつことになる．したがって手書き作図アプリケーション開発者はアプリケーション固有のユーザインタフェースの機能や入出力デバイスの機能を抽象的な機能モデルとして実装できることになる．

一方，仮想デバイス空間と論理空間に直接アクセスする仮想デバイス層の処理は本質的に各手書き作図アプリケーション間で共用化できる処理となっている．したがって仮想デバイス層をあらかじめライブラリ化しておけば，手書き作図アプリケーション開発者はこのライブラリを用いて共用化部分を実装し開発効率を向上させることができる．

4.2 MVCソフトウェアパターンによる仮想デバイス層の実装

仮想デバイス層をライブラリ化するにあたりMVCソフトウェアパターン[7]に基づいた実装を行うこととした．

MVCはオブジェクト指向言語SmalltalkでGUI設計に用いられたソフトウェアパターンであり，アプリケーションをModel, View, Controllerの3つの機能に分け，機能間のインタフェースを明確にして設計を行う手法である．MVCソフトウェアパターンに基づいた設計には，機能間の相互依存性が抑えられ，アプリケーション開発が見通しよく行えるようになるとい

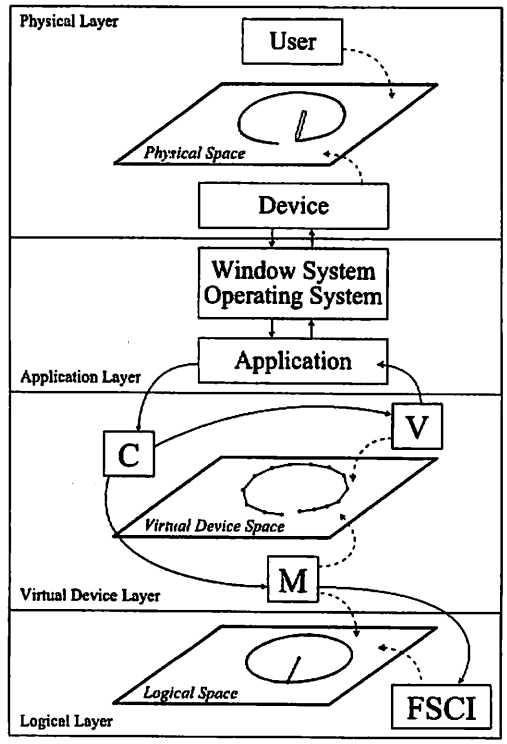

図7 仮想デバイス層を含む3層から構成される手書き作図アプリケーションと3つの空間との関係図

う特徴がある．ここでMVCソフトウェアパターンでは一般に，Modelはアプリケーションの中核となる論理的処理機能，Viewは表示出力をつかさどる機能，Controllerは ModelとViewを制御する機能として定義されている．

本稿ではMVCソフトウェアパターンに基づいて以下の方針で仮想デバイス層を機能分割することとした．

Model: 手書き図形認識エンジン FSCIに認識処理を委託しつつ，仮想デバイス空間と論理空間の両方にアクセスしてオブジェクトの管理を行う処理をModelとする．

View: 機能の抽象化された「仮想カメラ」として仮想デバイス空間中のオブジェクトにアクセスし，それらを画像としてアプリケーション層に出力する処理をViewとする．

Controller: 機能の抽象化された「仮想入力デバイス」としてアプリケーション層からの依頼に応じてModelとViewを制御する処理をControllerとする．

このように仮想デバイス層を構成することで，図7にみるように，アプリケーション層は仮想デバイス層のViewとControllerおよび物理層の現実デバイスとのみインタフェースをもつことになる．したがって，手書き作図アプリケーション固有のアプリケーション層を開発する作業は

- 仮想デバイス層の仮想入力デバイス (Controller) および仮想カメラ (View) と物理層の現実デバイスとの機能の関連付けを行う，
- 仮想入力デバイス (Controller) の機能および仮想カメラ (View) の機能だけを用いてユーザインタフェースを作成する，

という2つの作業に集約されるので，開発者は見通しのよい開発を行えることになる．

ここで物理層のデバイス環境に合わせてViewとControllerの機能を拡張する必要が生ずる場合もあるが，この場合でも開発者はViewとControllerの基本機能を継承拡張することで最小限の作業量により対応することができる．また，手書き作図アプリケーションの出力する最終的な認識図形の種類 (SKITの場合は幾何曲線列，BlueGrottoの場合は幾何立体図形) に応じてModelの機能を拡張する必要が生ずる場合もあるが，やはりこの場合でも開発者はModelの基本機能を継承拡張することで最小限の作業量により対応することができる．

以下では，Controller, View, Modelの各コンポーネントについて，より具体的な構成を示す．

4.3 Controllerの構成

Controllerを構成する仮想入力デバイスは以下の3種類に集約される．

描画デバイス: 描画を仮想デバイス空間中で表現可能な時系点列に変換し，その処理をModelに依頼する仮想デバイス

空間操作デバイス: 仮想デバイス空間と論理空間との間の位置関係を表す相似変換行列を制御し，その変換行列に応じた処理をModelに依頼する仮想デバイス

カメラ操作デバイス: 仮想デバイス空間中に存在する仮想カメラの位置・姿勢・視錐体を制御し，その位置・姿勢・視錐体に基づいた表示をViewに依頼する仮想デバイス

4.4 Viewの構成

Viewの機能はいわゆる仮想カメラの機能と一致する．この仮想カメラはその位置・姿勢・視錐体の設定に基づいて仮想デバイス空間中のオブジェクトを2次元画像化する．

4.5 Modelの構成

Modelの機能は仮想デバイス層の処理の中核となる論理的処理機能であるため，数多くの機能群から構成されることになる．以下にその代表的なものを列挙する．

(1) 仮想デバイス空間中の時系点列化された描画データを連続的な幾何曲線 (スプライン曲線) に変換し論理空間中に描画曲線として配置する機能

(2) 論理空間中のオブジェクトを点列データに変換し仮想デバイス空間中のオブジェクトとして配置する機能

(3) 手書き図形認識エンジンFSCIに論理空間中の描画曲線を幾何曲線列として認識するように依頼する機能

(4) 仮想デバイス空間中および論理空間中のオブジェクト間の関係性を管理する機能

5. 仮想デバイス層を導入した手書き作図アプリケーションの作成

4節で提案した仮想デバイス層の導入効果を確認するために，MVCソフトウェアパターンによる設計に基づいた仮想デバイス層をライブラリとして実装した．また，この仮想デバイス層ライブラリを用いて従来のSKITとBlueGrottoを3層構造でそれぞれ再構成した．

5.1 仮想デバイス層の実装

4節で提案したMVCソフトウェアパターンによる設計に基づいて仮想デバイス層をC++言語ベースのライブラリとして実装した．全体のステップ数は約4000ステップ程度となり，その内訳はControllerが約20%，Viewが約40%，Modelが約40%となった．

5.2 3層構造によるSKITの再構成

SKITを図8のように3層構造で再構成し，C++言語ベースのアプリケーションとして実装した．

ここでアプリケーション層の開発作業は以下の作業により完了した．

(1) 仮想入力デバイスの描画デバイスを物理層のタブレットのスタイラスと関連付ける作業

(2) 仮想入力デバイスの空間操作デバイスをSKITのクライアント作図アプリケーションの論理空間情報と関連付ける作業☆

(3) 仮想入力デバイスの描画デバイスだけを用いた幾何曲線入力ユーザインタフェースを作成する作業

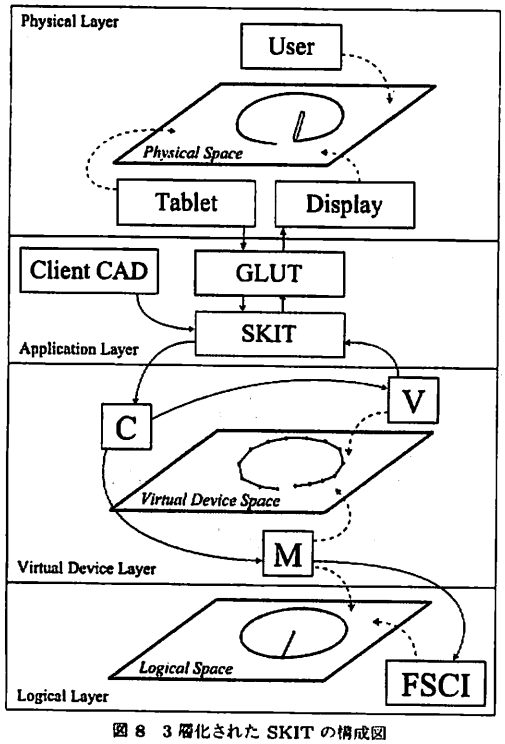

図8 3層化されたSKITの構成図

☆ SKITはクライアント作図アプリケーションの論理空間情報を取得して自らの論理空間をクライアントに同期させる．そのためここではクライアント作図アプリケーションを物理層のデバイスとしてみなした実装を行う．

なお，SKITでは視点が固定されているので仮想入力デバイスのカメラ操作デバイスに対する関連付け作業は行っていない．

一方，仮想デバイス層の開発作業に関しては，SKITの最終認識結果がFSCIが認識する幾何曲線列に限られるため，継承拡張作業の必要性は発生しなかった．

再構成したSKITのソースコードのステップ数を調査した結果，論理層のFSCIは約7000ステップ，仮想デバイス層のライブラリは約4000ステップ，アプリケーション層のSKITは約2800ステップとなった．このことから，3層構造のSKITにおける共用化部分(FSCIおよび仮想デバイス層ライブラリ)の割合は約80%に達することを確認できた．これは2層構造のSKITにおける共用化部分(FSCI)の割合である約50%と比較して大幅に増加しており，本稿で提案した仮想デバイス層の導入がSKITの開発効率改善に大きく寄与することがわかる．

5.3 3層構造によるBlueGrottoの再構成

BlueGrottoを図9のように3層構造で再構成し，C++言語ベースのアプリケーションとして実装した．

ここでアプリケーション層の開発作業は以下の作業により完了した．

(1) 仮想入力デバイスの描画デバイスを物理層の3

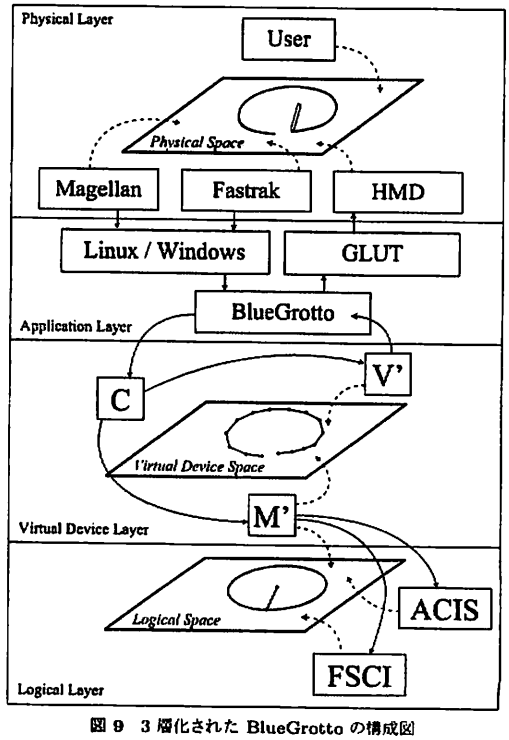

図9 3層化されたBlueGrottoの構成図

次元位置センサ (Fastrak センサ) 搭載スタイラスと関連付ける作業

(2) 仮想入力デバイスの空間操作デバイスを物理層の 3 次元コントローラ (Magellan) と関連付ける作業

(3) 仮想入力デバイスのカメラ操作デバイスを物理層の HMD に取り付けた 3 次元位置センサ (Fastrak センサ) と関連付ける作業

(4) 仮想入力デバイスの描画デバイスと空間操作デバイスを用いた幾何立体図形入力ユーザインタフェースを作成する作業

一方，アプリケーションの開発に伴って必要となった仮想デバイス層の継承拡張作業は以下のとおりとなった．

(1) BlueGrotto の最終的な認識図形が幾何立体図形であるため，幾何曲線列の組み合わせとして幾何立体図形を管理できるように Model を継承拡張する作業

(2) 幾何曲線列から幾何立体図形を生成する処理を幾何エンジン (ACIS[8]) に依頼できるように Model を継承拡張する作業

(3) BlueGrotto のステレオ HMD に対応するために，一体動作する仮想カメラペアの機能を実現するように View を継承拡張する作業

再構成した BlueGrotto のソースコードのステップ数を調査した結果，論理層の FSCI は約 7000 ステップ，仮想デバイス層のライブラリは約 4000 ステップ，BlueGrotto の仮想デバイス層の継承拡張部分は約 500 ステップ，BlueGrotto のアプリケーション層は約 2700 ステップ，となった．このことから，3 層構造の BlueGrotto における共用化部分 (FSCI および仮想デバイス層ライブラリ) の割合は約 77%に達することを確認できた．これは 2 層構造の BlueGrotto における共用化部分 (FSCI) の割合である約 50%と比較して大幅に増加しており，本稿で提案した仮想デバイス層の導入が BlueGrotto の開発効率改善にも大きく寄与することがわかる．

6. まとめ

本稿では，汎用的手書き図形認識エンジン FSCI を利用した手書き作図アプリケーションを従来よりもさらに見通しよく効率的に開発できるようにすることを目的とし，アプリケーション層と論理層の中間に新たに MVC ソフトウェアパターンに基づいた仮想デバイス層を導入することを提案した．また，仮想デバイス層を C++言語ベースのライブラリとして実装し，このライブラリを利用して従来の 2 種類の手書き作図アプリケーション SKIT および BlueGrotto を 3 層構造で再構成した結果，これらの手書き作図アプリケーションの開発を見通しよく行えるという知見を得た．さらに手書き作図アプリケーションでのソースコードの共用化が大幅に進み開発効率が向上することも示した．

今後，さらに多様な手書き作図アプリケーションの開発に今回提案した仮想デバイス層を用いた 3 層構造の設計を適用することで，仮想デバイス層ライブラリの有効性を確認するとともに，その機能の洗練を図る予定である．

参考文献

1) 佐賀聡人，牧野宏美，佐々木淳一：ファジースプライン曲線同定法，電子情報通信学会論文誌，Vol.J77-D-II, No.8, pp.1620–1629 (1994).

2) Saga, S. and Makino, H.: Fuzzy spline interpolation and its application to online freehand curve identification, *Second IEEE International Conference on Fuzzy Systems (Cat. No.93CH3136-9)*, pp.1183 – 90 (1993).

3) 河合良太，西川　玲，佐賀聡人：手書きスケッチ入力フロントエンドプロセッサ:SKIT，電子情報通信学会論文誌，Vol.J88-D-II, No.5, pp.897–905 (2005).

4) 井上智之，西住直樹，鈴木伸明，安福尚文：仮想空間中での手書きジェスチャ認識に基づいた 3 次元モデリング，電子情報通信学会論文誌，Vol.J87-D-II, No.6, pp.1309 – 1318 (2004).

5) 安彦健之介，佐賀聡人：3 次元手書き図形入力インタフェース BlueGrotto へのスキニング機能の導入，画像電子学会 研究会講演予稿，Vol.226, pp.25–28 (2006).

6) 佐藤洋一，安福尚文，佐賀聡人：スケッチによる作図インタフェースのための逐次型ファジースプライン曲線生成法，電子情報通信学会論文誌，Vol.J86-D-II, No.2, pp.242 – 251 (2003).

7) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*, Addison-Wesley Pub (Sd) (1995).

8) Corney, J. and Lim, T.: *3D Modeling with ACIS*, Saxe-Coburg Publications (2001).