6F－4　オブジェクトネットワークによる画像システム記述言語

鴨志田　稔　　丹羽直人　　榎本　肇

芝浦工業大学

1．はじめに

システムの設計において、設計者とユーザの両方の立場から、データを中心にシステムを構築するのが望ましい。システム内部での処理を、データの変換過程として捉え、その変換を起こす関数が存在すると考える。ここで、データの変換の各段階をノードとし、関数をブランチとすると、これをネットワークとみなせる。このネットワークをオブジェクトネットワークと呼ぶ。

オブジェクトネットワークにより設計されたシステムをインプリメントする場合、全てC言語などの手続き型汎用言語で行うことは効率が悪い。そこで、データと関数の定義し、オブジェクトネットワークを構築するシステム記述言語が必要である。この言語は定義する全てのものをオブジェクトとみなし、マルチウインドにより表記される。

この論文では、ウインド表記法を用いた画像システム記述言語WELL-PPP[4](Window-based ELaboration Language for Picture Processing and Painting)に、新たにオブジェクトネットワークの概念を加えたWELL-PPP Ver 2.0について述べている。WELL-PPP Ver2.0（以下、WELL v.2と省略する）は、分野記述型言語[1]の一種であり、画像描画にカスタマイズした例である。

言語の応用例を画像描画システムという、一般的に理解しやすい分野に応用することにより、この言語の概念、構造、そして結果について説明する。

2．オブジェクトネットワークと画像描画

2．1．画像データ

画像データは、画像データクラスのインスタンスとして階層的に定義される。WELL v.2のデータクラスは、そのデータクラスそのものを表す”項目”とそのデータクラスの性質や特徴を表す”属性”を持つ。次に示す画像データクラスも項目と属性に分けて定義されている。画像データは、3章で説明するデータウインドにより表記される。表1に示す画像データクラスは、アプリケーションである画像描画システム用に定義されたものであり、一般的に定義されたものではないことを断っておく。

2．2．描画関数

WELL v.2では、あるノードに対応する画像データを次のノードに変換する関数を具体的描画関数として定義する。具体的描画関数はノード間をつなぐ各ブランチに対して定義される。具体的描画関数の集合としての関数族を総称的描画関数と考える。総称的描画関数内の各具体的描画関数には、その関数が実行される制約条件が付加される。ユーザが総称的描画関数を指定することにより、システム側が現在の状態から対応する具体的描画関数を選び出し実行する。以下、具体的描画関数をActual関数、総称的描画関数をGeneric関数と呼ぶことにする。

関数実行の駆動方法は、イベント駆動とデータ駆動の2つがある。イベント駆動は、ある事象（イベント）が発生したとき、その事象の種類により、関数が選択され実行する方式である。データ駆動は、イベント駆動によりリクエストされた関数の実行に必要なデータが存在しない場合、そのデータを生成する関数を実行する方式である。よって、データ駆動はイベント駆動と対になって使われ、単独で使われることはない。

System Description Language for Picture Processing and Painting based on Object Network

M.Kamoshida, N.Niwa, H.Enomoto
Shibaura Institute of Technology

2．3．オブジェクトネットワーク

画像データをノードとし、ノード間をつなぐGeneric関数をブランチとすることにより、図1に示す描画用オブジェクトネットワークが構成される。オブジェクトネットワークはオペレーションウインド（3章で説明）に表記される。

オブジェクトネットワークは、データの項目に対する処理を行う項目ネットワークとそのデータの属性に対する処理を行う属性ネットワークに分けられる。描画用オブジェクトネットワークでは、項目ネットワークとして"Frame"、属性ネットワークとして"Color"が定義されている。処理対象である画像データが、"Frame"ネットワークに沿ってNONEからDESSIN_ELEMENTまで変化することにより、画像の輪郭が描かれる。そして、"Color"ネットワークに沿って、画像データに色度・輝度データを与えることで、色度・輝度描出が行われる。つまり、ネットワーク上を移動することが、画像描画を行うことに等しい。画像処理は、オブジェクトネットワークを上から下への移動に対応し、画像データは変わらないが、処理関数は描画関数の逆関数という形で定義できる。ここに、画像処理と画像描画の双対性を見ることができる。

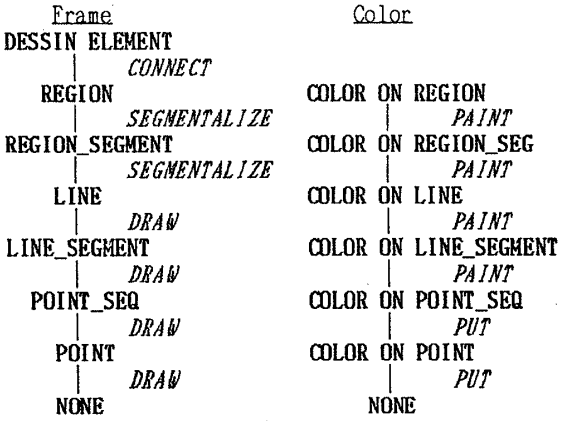


図1　描画用オブジェクトネットワーク

3．WELL-PPP Ver2.0の構造

WELL v.2は、言語を実行形式に変換するトランスレータと実際に実行を行うランタイムシステムから構成される。トランスレータは、WELL v.2の文法により記述されたオブジェクトネットワークやウインドの定義を、ランタイムシステム用の実行形式データに変換し、定義された関数を汎用言語に変換し、ランタイムシステムとリンクされる。ランタイムシステムは、コンパイラにより用意されたデータを用いて、ユーザからのリクエストに従い、関数の実行を行う。図2は、ランタイムシステムの構造を表したものであり、これについて説明する。

3．1．WELLカーネル

WELLカーネルは、ランタイムシステムの中心に存在し、トランスレータからのデータを基に、

A.オブジェクトネットワーク上のデータ状態把握
B.イベントの解析
C.関数実行要求

を行う。A.は、描画プロセスの把握を行い次に実行すべき関数を選択する。B.は、各ウインドから送られてくるイベントを判別し、その結果に対応するシステム内のプロシジャをコールする。C.は、Generic関数に制約条件を付加して、3．4で説明するオブジェクト間インタ

フェースに関数の実行をリクエストする。

3．2．マネージャ

WELL v.2では、データ、関数、ウインドをマネージャにより一括管理を行っている。これにより、イベント駆動、データ駆動による関数実行メカニズムを実現している。ランタイムシステムで使われるマネージャは、

A.データマネージャ
B.関数マネージャ
C.ウインドマネージャ

がある。A.は、画像データの管理を行う。B.は、関数の実行管理を行う。そして、C.は、ウインドの管理、ウインドへのオブジェクトの描画を行う。これらのマネージャは、カーネル、Actual関数、または他のマネージャから、オブジェクトインターフェースを通してリクエストーレスポンド方式のプロトコルにより利用される。マネージャは、構築するシステムによりカスタマイズされるが、そのプロトコルは変更されないので、他に影響を与えずカスタマイズか可能になる。

3．3．ウインド

WELL v.2では、全てのオブジェクトは、ウインドにより表記されるべきであるとし、次の4種類のウインド(図3)を利用できる。

A.オペレーションウインド
B.データウインド
C.メッセージウインド
D.選択ウインド

A.は、オブジェクトネットワークを表示し、ユーザからの関数リクエストを受け付ける。また、ランタイムシステム全体のオペレーションを行う。B.は、画像データを表示し、座標データの入力を受け付ける。C.は、ユーザに対してメッセージを表示するために使われる。D.は、ユーザによる候補選択のために使用され、メニュータイプとキータイプが定義できる。

3．4．オブジェクト間インターフェース[2]

オブジェクト間インターフェースは、WELLカーネルとデータ、関数、ウインドを管理するマネージャ間を、リクエストーレスポンド方式の標準的プロトコルによりインターフェースをとる。標準的プロトコルを用意することで、カーネル部には変更を加えずに言語のカスタマイズが可能になる。オブジェクト間インターフェースのプロトコルは、データ、関数、ウインドマネージャに対しての3つに分けられる。データマネージャに対しては、読み書き、変更、削除などデータ操作に関するプロトコルが用意されている。関数マネージャに関しては、Generic関数からActual関数への変換、Actual関数の実行に必要とするデータの準備、Actual関数の実行要求プロトコルがある。そして、ウインドマネージャへは、データウインド、選択ウインド、メッセージウインド、オペレーションウインドに対するプロトコルが用意されている。

4．結果

今回は、WELL v.2の開発を、SPARC Station2/GSで行い、インプリメンテーションを簡単にするために、単一プロセス（直列）で行った。しかし、これは、よりインプリメンテーションを複雑にしてしまった。なぜなら、図2に示すように、ランタイムシステムの各モジュールが、並行に動作することを前提に設計したためである。よって、次のバージョンでは、各モジュールを一つのプロセスとして、並行システムとしてインプリメントすべきである。直列に行うより、スムーズにインプリメンテーションが行えるであろう。

5．まとめ

図1に示した、描画用オブジェクトネットワークは、画像の個別対象を生成し、"Frame"ネットワークと"Color"ネットワークの独立性から、それぞれの関数を並行処理で実行できる。このネットワークより一つ上位のネットワークとして、個別対象を組み合わせる構造ネットワークが必要である[3]。構造ネットワークは、個別対象の組み合わせ方を表し、個別対象は、複数のユーザに並行に生成される。つまり、画像描画において、並行処理は以上の様な二つのレベルが存在する。

並行処理の実現には、オブジェクト間インターフェースが大変重要である。個別対象の並行処理の場合には、要求された複数の関数が、並行に実行可能であるか、ネットワークから判別され、各関数を実行する適当なリソースを割り当て、関数マネージャがそれらを実行する。また、構造ネットワークの並行処理では、複数のユーザにより、複数の描画用ネットワークの実行が要求され、各ネットワークに適当なリソースが割り当てられる。よって、ネットワークにより接続された複数のワークステーションを利用して、画像描画用CSCWシステムの実現が行える。

並行処理にWELL-PPP Ver2.0を用いることで、ユーザフレンドリな新しい方式の並行システムの実現が可能となるだろう。

文献

[1]榎本、鴨志田：" 分野記述言語の構造"，情報処理学会第44回全国大会，1992.3
[2]丹羽、鴨志田、榎本：" オブジェクトネットワーク上の実行処理でのインターフェースとプロトコル"，情報処理学会第44回全国大会，1992.3
[3]守屋、鴨志田、村尾、榎本：" オブジェクトネットワークを利用した並行システムのペトリネット表現"，情報処理学会第44回全国大会，1992.3
[4]鴨志田、榎本、宮村：" カラー画像処理・描画用ウインド型言語「WELL-PPP」"，情報処理学会第42回全国大会，4Q-1，1991.3

表1　画像データクラス

画像データクラス	項目	属性	意味
COORDINATES	変数（x，y）	入力されたデータウインドデータウインド名	座標を表す
POINT	COORDINATESにより示された座標	輝度データ、色度データ	ポイントを表す
POINT_SEQ	POINTの順序集合	輝度データ、色度データ	POINTの並びを表す
LINE_SEG.	接続するPOINT_SEQ LINE_SEGを構成するPOINTの集合	輝度データ、色度データ、接続形状	POINT_SEQを接続する最小単位
LINE	接続可能なLINE_SEGの集合	輝度データ、色度データ	ラインを表す
REGION_SEG.	REGION_SEGの始点と終点 挟む二つのライン REGION_SEGを構成するPOINTの集合	輝度データ、色度データ	二つのラインに挟まれた一走査線分の領域
REGION	REGION_SEGの集合	輝度データ、色度データ、前後関係	連続するREGION_SEGの集合
DESSIN_ELEMENT	REGIONの集合	前後関係	REGIONの集合

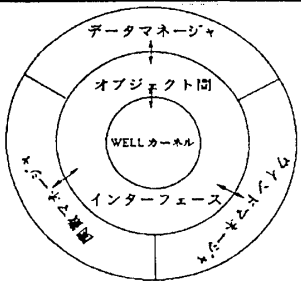
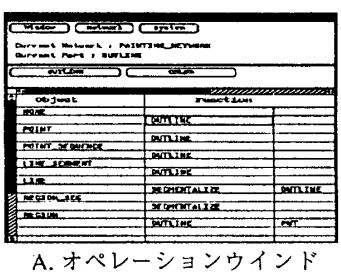
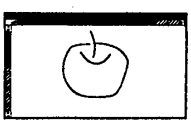


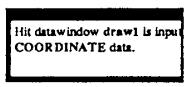
図2 ランタイムシステムの構造



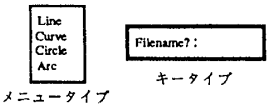
A.オペレーションウインド



B.データウインド



C.メッセージウインド



D.選択ウインド

図3 ウインド