日本包装学会誌 Vol. 3 No. 4 (1994)

一般論文

オブジェクト指向による三次元配置決定システム

島田哲夫* 清水淳也* 山崎潔**

3D-Loading Pattern Determination System Based on Object-oriented Approach

Tetsuo SHIMADA*, Junya SHIMIZU*, Kiyoshi YAMASAKI**

This study aims to develop a loading pattern determination system based on object oriented approach. With respect to loading pattern determination problem,there are three important factors: (1) the loading expert technique of packages in the field of distribution, (2) optimal pattern determination algorithms, (3) information system. The authors have already developed a loading pattern determination system including the above two factors: (1) the loading expert technique and (2) optimal algorithms.

As described in this report, there have been a remarkable increase in the use of strategic information about management as well as the expert knowledge about packaging and distribution. Then we intended to derive a scheme to develop a loading pattern determination system based on object -oriented approach as the next step.

Keywords : Information system, Object-oriented approach, Loading pattern, Optimal pattern design, Three dimensional pattern

一定のパレット上に包装品を積み付ける方法は、包装設計および物流設計の両方に関わる重要な課題である。本論文では積み付けパターンを自動決定するための包装・物流設計に的を絞った「オブジェクト指向による三次元配置決定システム」について記述する。包装・物流設計に関するソフトウェアは、現在よりきめの細かい機能を求められつつあるため、設計情報が増加する一方、経営戦略に絡んだ他部署の情報も包装技術に加味されるようになってきている。そこで、画面上に多くのデータを表示させながら設計に不可欠な繰り返し入力による最適解の導出作業を効率よくおこない、さらに設計技術の進展にともなって大規模化する傾向にあるソフトウェアの維持管理の効率化のため、抜本的なプログラミング手法の変更が必要となる。

本稿では、今後包装・物流分野のシステム開発の課題であるマンマシーンインタフェースの改善を図り、かつ大規模システム化を可能とするマルチウィンドウシステムについて検討し、積み付けパターン配置決定システムを構築した。

キーワード：情報システム、オブジェクト指向、最適パターン設計、3次元パターン

* 流通科学大学（〒651-21 兵庫県神戸市西区学園西町3-1）：University of Marketing & Distribution Sciences, 3-1, Gakuen-nishimachi, Nishi-ku, Kobe-shi, Hyogo, 651-21 **兵庫県立工業技術センター（〒654 兵庫県神戸市須磨区行平町3-1-12）：Hyogo Prefectural Institute of Industrial Research, 3-1-12, Yukihira, Suma-ku, Kobe-shi , 654

1. 緒　言

包装・物流設計に関する重要な問題として、一定パレット上に包装品を規則正しく配置する問題がある。この問題には (1) 包装・物流技術に関する要素、(2) 最適配置というアルゴリズムに関する要素、(3) 情報システム構築というソフトウェア開発に関する要素等が互いに絡まっているため、容易に解決を図ることが難しい課題がある。そこで筆者らは積み付けパターンに関する三次元配置問題を解くため、包装物流分野での制約条件を整理し、積み付け作業における各種条件と積載効率の最適性とのトレードオフから現状に沿った方法を提案した[1)~3)]。本稿では (3) 情報システムを構築するためのソフトウェア開発に関する要素からこの問題を考える。

従来包装・物流設計に関するソフトウェアは、一定の決まった手続きにしたがい、順序よくデータを入力していくことにより、一つまたは複数の結果を導出していた。設計に必要な包装・物流に関する知識やデータが少なくて済み、かつ最適な設計に至るまでの繰り返しが比較的少ない作業では従来のプログラミング方法で十分である。しかし今後ますます多様化が進み、設計時点できめの細かい情報が必要となるため、包装設計に関する各種データは増加する方向にあり、さらに経営戦略に絡んだ他部署の情報も加味されようとしている。

そこで、画面上に多くのデータを表示させながら操作性よく入力でき、しかも設計に不可欠な部分データの繰り返し入力による最適解の導出作業を効率よくおこなえるようなシステムを構築することが今後の課題となっている。さらに設計技術の進展にともなって規模が拡大する傾向にあるソフトウェアの維持管理のため、抜本的なプログラミング手法の変更が必要となる。

本論文では、今後包装・物流分野のシステム開発の課題であるマンマシーンインターフェースの改善と大規模システム構築のためオブジェクト指向による三次元配置決定システムについて記述する。

オブジェクト指向の言語仕様[4) 5)]は、物や概念など世の中にあるものをオブジェクト（データとそれを処理する関数のまとまり）であると定義することからはじめる。さらに機能と属性を含んだ同じオブジェクトの抽象概念をクラスという。その特徴はカプセル化（オブジェクト内部のデータを外部から直接アクセスできないようにすること）、継承（似た性質をもつクラス間の結合の方法）、多態性（少しずつ異なっているが、似通った機能を多重に定義するすることを可能とすること）の3つに集約することができる。またオブジェクト指向プログラミングの利点は、現実の世界を克明にモデル化すれば、オブジェクト間でのメッセージの受け渡しを記述するのみで事態が進行していくことにある。すなわち、規格化することを念頭において言語仕様が規定されており、しかも規模の拡大に必然的に伴ってくる複雑さを解消するため、部品間のつながりをスムーズにおこなうことができるように工夫されているのである。

本稿の三次元配置決定システムはマルチウィンドウをおもな機能としている (Fig. 1)。Microsoft－WINDOWSやMacintosh－OSをはじめとして、マルチウィンドウは今後のマンマシーンインターフェースの主流であ

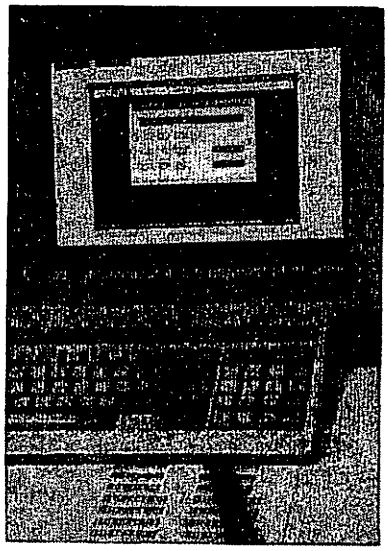


Fig. 1 Loading pattern determination system

る。その理由として、ウィンドウを用いることにより、画面をより広く使うことおよび画面を見やすくすることが容易に可能となる利点をあげることができる。しかし、そのような利点の反面、プログラマが管理する情報はウィンドウを用いないシステムと比べると一般に多くなる。ウィンドウを表示するためには、まずウィンドウの大きさ、ウィンドウのタイトル、そしてウィンドウがオーバーラップするときには背景のデータ等を管理しなくてはならず、ウィンドウの数が少数であればいいが、これが多くなってくると、プログラマ自身が管理するのは困難になってくる。また既存のマルチウィンドウシステムは汎用性を重視するあまり、たとえば包装・物流業務に特化した処理を構築することが不得手であると言われている。

このようにウィンドウシステムの利点を生かし、かつ既存のウィンドウシステムの欠点を克服するためには、包装・物流業務に沿った操作性のよいマルチウィンドウ機能をもつ業務処理システムを独自に構築する必要がある。

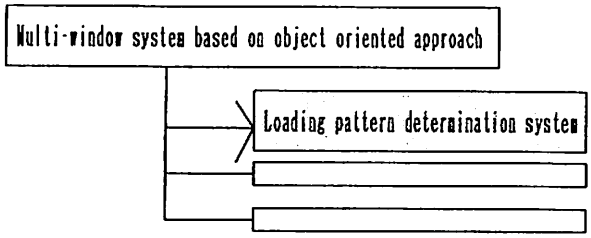


Fig. 2 System Configuration

そこで本稿の目的は、ウィンドウをオブジェクトとして部品化することにより、包装・物流技術に適合したマルチウィンドウ機能を有する3次元配置決定システム (Fig. 2) を作成することである。

なおハードウェアはNECの9801FA (メモリ4MB) を用い、MS－DOSのVer.5.0上で稼働するオブジェクト指向言語TurboC^{++}を用いて構築することとする。

2. マルチウィンドウシステムの構成

オブジェクト指向プログラミングの基本は、一定のデータとその処理関数をまとめたオブジェクトを単位として構成することにある。

本ウィンドウシステムの構成要素は、アイテムタイプ (本システムを構成する個々の要素) とマネージャタイプ (本システムを管理する要素) のクラス (オブジェクトの抽象概念) に大別することができる。またアイテムタイプはさらにアイコン・ボタンタイプと、ウィンドウタイプ、コレクションタイプに分けることにする。Fig. 3はマルチウィンドウ

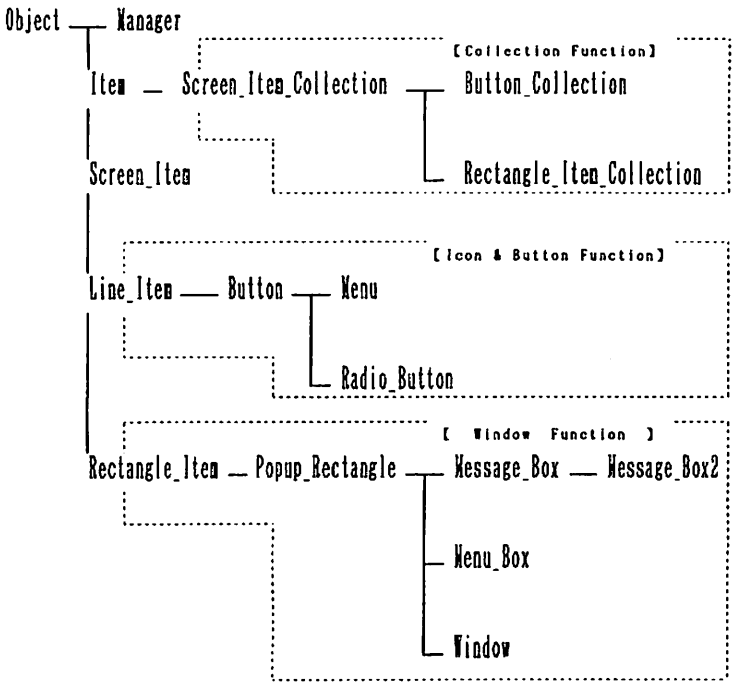


Fig. 3 System Sequence

システムにおける全クラスの関連図である。Managerクラスは全てのクラスのオブジェクトをScreen_Itemクラスのオブジェクトとして管理している。

Managerクラスの主な機能は、管理するオブジェクトに対して、

(1) 入力情報の受け入れを許可する
(2) 入力情報の受け入れを許可しない
(3) アクティブ状態にする
(4) 非アクティブ状態にする
(5) 外部からの入力情報（キーボード、マウスからの入力）を与える
(6) 画面におけるオブジェクトの位置を変えるなどの指示だけを与え、いわば交通整理のみをおこない、それに対して

(1) メッセージ（指示）を解釈する
(2) メッセージの内容を実行する

等々はオブジェクト単位に任せている。すなわち、実際の動作はクラスを実体化したオブジェクトが自分のなかの関数を使って実行するようにしている。以下、それぞれの詳細については、アイコン・ボタン、ウィンドウ、コレクションの3つの構成に分けて記述する。

2.1 アイコン・ボタンの構成

マルチウィンドウシステムではマウスで画面の特定の部分をクリックしたり、ドラッグしたりすることで、ユーザーは好みの機能を選択することができる。この機能を実現するためには

(1) 画面上のある特定部分に機能をイメージで表したアイコン、または機能を文字で表したボタンを配置する
(2) それらに対してユーザーがマウスでクリックしたり、ドラッグしたときは、表示している内容に対して適切な処理を行う

等が必要となる。

Fig. 4は本システムで構築したアイコン・ボタンタイプクラスの継承関係である。各々

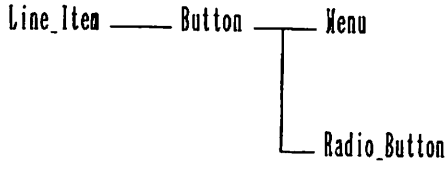


Fig. 4 Sequence of button type class

のクラスの機能は以下のとおりである。

(1) Line_Item クラス
- ボタン、アイコンタイプクラスの基本となる抽象クラス。
- 基本クラスであるScreen_Itemクラスより外部とのインターフェースだけを継承。
- 画面上における3つの座標を保持している。

(2) Button クラス
- 一般的なボタンクラス。
- 外部からの入力情報をオブジェクトが保持している情報と比較し、適切な入力なら一定の反応を示す。

(3) Menu クラス
- 画面上部に並ぶプルダウンメニューのためのクラス。
- 外部から適切な入力情報を受けるとプルダウンメニュー等を開く。

(4) Radio_Button クラス
- 基本的にはButtonクラスの機能と同じであるが、このクラスは外部から入力情報毎に True、Falseのふたつの状態を交互に変えながら保持している。

本システムのアイテムタイプのクラスのすべてのもととなるクラスであるScreen_Itemクラスから Line_Itemクラスを導き出す。これによりScreen_Itemクラスの画面上のある一点の情報を管理するという機能に、新たに画面上もう一点の座標を管理する機能を付け加えたことになる。さらにLine_Itemクラスから、画面にボタンを表示する、マウスからの入力情報を処理し、それに対して処理結果を返す、等々の機能を付け加えたButtonクラスを導き出している。このButtonクラスの関数の機能は、入力情報を受け取りそれに対しての処理、入力情報の受け取りを許可、ボタンを画面に表示、ボタンの画面上の表示位置を変更、などがあり、これらの関数のインターフェース規約は継承もとであるScreen Itemクラスのものと同じである。このようにインターフェース規約が決まっているということはオブジェクトをScreen_Itemクラスのオブジェクトとして管理しているManagerクラスや、後に説明するコレクションにとってはきわめて重要なことである。つまり、インターフェースが決まっていればマネージャーやコレクションはオブジェクトがどんなクラスのオブジェクトであれ、Screen_Itemクラスから継承されたクラスのオブジェクトである限り、それに対して決まった処理を行うだけですみ、システムが拡張、変更されてもそれに対して柔軟に対応できるからである。

2.2 ウィンドウの構成

マルチウィンドウシステムの主要機能について以下に述べる。ウィンドウを用いることにより、画面をより広く使うことおよび画面を見やすくすること、が容易に可能となる。

Fig. 5は本システムで構築したウィンドウタイプクラスの継承関係である。各々のクラスの機能は以下のとおりである。

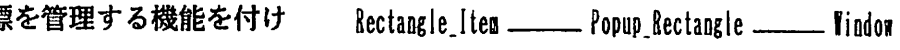


Fig. 5 Sequence of window type class

(1) Rectangle_Item クラス
- ウィンドウタイプの基本となる抽象クラス。
- Line_Itemクラスより外部とのインターフェースと画面上の3つの座標を保持という機能を継承。

- 画面上における４つの座標を保持。

(2) Popup_Rectangle クラス

- ウィンドウタイプのクラスはオーバーラップが出来ることを前提としている。このクラスではオブジェクトが生成される前にオバーラップ領域を待避し、オブジェクト消滅時にはこれをもとに戻している。

(3) Window クラス

- オーバーラップができるほか画面内で自由にその位置を変えたりサイズを変更できる。
- このシステムではこのウィンドウを単位としてシステムを作ることを前提としている。

基本となるWindowクラスでウィンドウ内のアイコン・ボタンが押されたときには主たる機能を定義した関数を呼び出すようにしている。この関数に多態性を持たせているためWindowクラスを継承したクラスにおいて、関数をそれぞれに定義することにより、基本となる部分を変えないでアイコン・ボタンがクリックされたときの処理をおこなうことができる。

2.3 コレクションの構成

本マルチウィンドウシステムは多数のオブジェクトによって構成されている。オブジェクトの数が多数になってくると各々のオブジェクトをプログラマが管理・統合するのは困難になってくる。この困難を解決するために必要となってくるのがコレクション、コンテナ・クラスという概念である。つまりオブジェクト一つ一つをプログラマが管理するのではなくコレクションという管理・統合専門のオブジェクトにそれを任せるというものである。こうすることによりプログラマは多数のオブジェクトの管理に悩まされるということを避けることができるのである。

Fig. 6は本システムで構築したコレクションタイプクラスの継承関係を示している。各々のクラスの機能は以下のとおりである。

(1) Screen_Item_Collection クラス

- コレクションタイプのクラスの基本となる抽象クラス。
- 外部から登録されたオブジェクトを管理・統合する。

(2) Button_Collection

- ボタンタイプのクラスのオブジェクトを管理・統合する。

(3) Menu_Collection

- プルダウンメニューオブジェクトを管理・統合する。

(4) Rectangle_Item_Collection

- ウィンドウ、メッセージ・ボックスなどのウィンドウタイプのクラスのオブジェクトを管理・統合する。

Screen_Item_Collectionクラス、その派生

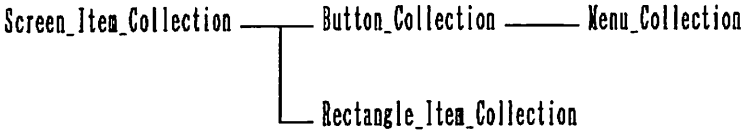


Fig. 6 Sequence of collection type class

クラスであるButton_Collectionクラス、Rectangle_Itemクラスなどがコレクションクラスである。まずコレクションタイプのクラスの基本クラスであるScreen_Item Collectionクラスには、管理するオブジェクトを登録、そして管理しているオブジェクトを消去などがあり、これら関数を使用してプログラマはオブジェクトの管理をコレクションクラスに任せることができる。ここで各オブジェクトはリスト構造をとって管理するようにしている。登録されるべきオブジェクトの数は不特定多数であり、個数を限定すれば汎用性を失わせる結果となるためであり、リスト構造にしておくと、オブジェクトが生成する度にリスト化することが可能となるからである。

3. 配置決定システム

3.1 手続き指向とオブジェクト指向の比較

上述したマルチウィンドウシステム上に積み付けパターンのための配置決定システムを構築することにする。

積み付けパターンは、対象とする製品の大きさおよび重量等により異なり,規格化することは困難である。包装品の大きさに関するパラメータである縦・横・高さと、パレットの広さに関するパラメータである縦方向の有効長および横方向の有効長などの相互関係に、規則性を加味した積載効率から積み付けパターンを決定しなければならない。

しかも、内容品によっては、外装寸法のパラメータを変更することが可能なものも含まれており、修正入力による結果の照合を繰り返し、最適な包装設計に収束させるためのシステムづくりが必要となる。

マルチウインドウシステムを用いて積み付けパターンという3次元配置決定システムを実現する場合の処理の概略の比較である(Fig. 7)。Fig. 7 (1) は従来からの積み付けパターン決定のための手順である。まず包装品の縦 (L)、横 (W)、高さ (D) を入力する。さらにパレットの縦に余裕寸法 (La) を差し引いた縦方向の有効長 (Lp − La) と、パレットの横 (Wp) に余裕寸法 (Wa) を差し引いた横方向の有効長 (Wp − Wa)、および積み段数 (dn) を入力する。このようにして包装品の縦、横、高さでパターンを構成したときの全領域が、有効なパレットの領域内になるかどうかを判定する。これを満たし、段当りの積み段数の最も大きなものを最適な積み段数 (Np) とする (なお倉庫高さ (Hmax)、パレット高さ (Dp) は一定とする)。ここで積載効率は

(1) 面積効率 (A_{eff})

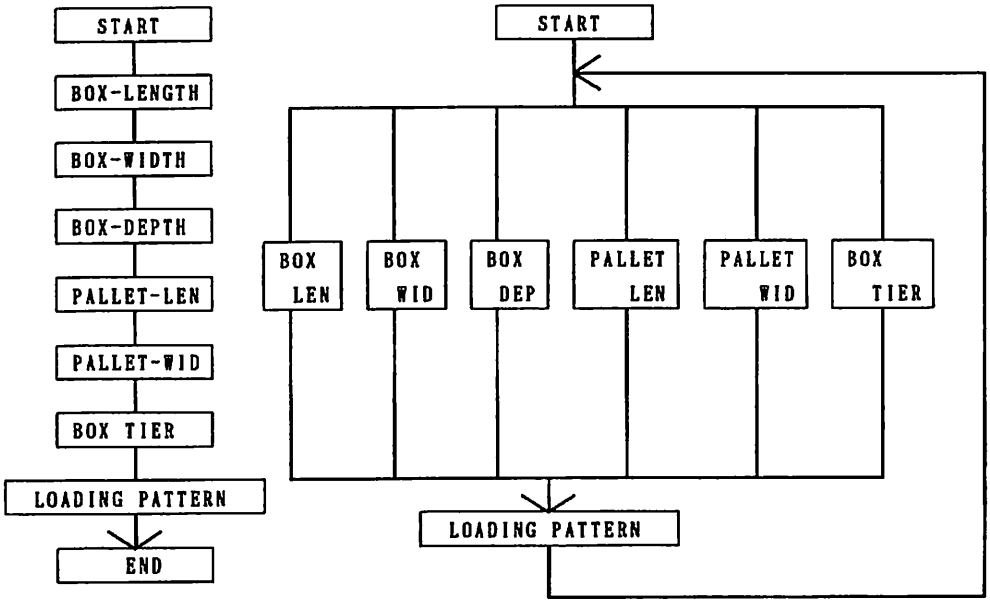
$$A_{eff} = \frac{L \cdot W \cdot Np \cdot 100.0}{(Lp - La) \cdot (Wp - Wa)}$$

(2) 体積効率 (W_{eff})

$$W_{eff} = \frac{A_{eff} \cdot (D \cdot dn + Dp) \cdot dp}{H_{max}}$$

となる。

従来の手続き型の処理では、処理開始から処理終了まで一方向の流れのため、基本的には、一組の入力に対して一つの結果が得られて終了する。しかし、最適な設計結果を得るためには、必要となる一部のデータのみ修正入力し、繰り返し作業により、設計結果を収束させる必要がある。 Fig. 7 (2) はオブジェクト指向に基づく処理内容である。 従来型の処理では入力すべき順番が決まっている



Fug. 7 Loading pattern determination flow

ため、その順に従って処理が進行し、積み段数のみ変更したいとか、パレット横のデータを変えたいという柔軟な設計作業に不可欠な修正作業の操作に不向きである。逆に本システムでは一度すべての値を入力し、表示された図と積載効率の値を見ながら積み段数のみ変更したいとか、パレットの横の値を、最適な値になるように微調整をおこなうといった繰り返し作業が可能となる。

3.2 アイコン・ボタン機能

オペレータとシステムとの会話（マンマシーンインターフェース）の基本としてアイコン・ボタン機能は重要である。システム環境設定時や終了時点などはいうに及ばず、インタラクションがおこる箇所はすべて、アイコン・ボタン機能が必要である。

Fig. 8は数値入力時の表示画面である。数値入力には（0から9）までの数値および(BS) バックスペースなどのボタンが必要である。それぞれのボタンを押下することにより対応する数値等を選択することができる。

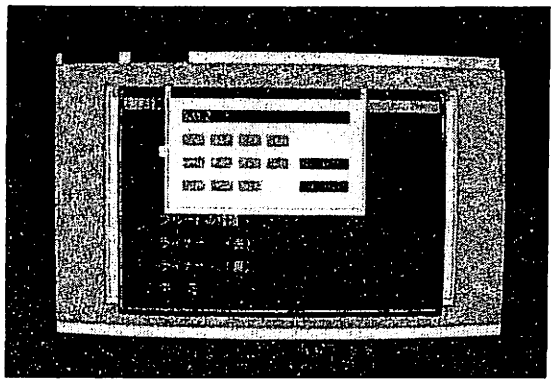


Fig. 8 Icon & Button function

3.3 マルチウィンドウ機能

基本となるWindowクラスでウィンドウ内のアイコン・ボタンが押されたときにはこのウィンドウに対応する関数が呼び出されるようにしている。この関数に多態性を持たせているためWindowクラスを継承したクラスにおいて、関数をそれぞれに定義することにより、基本となる部分を変えないでアイコン・ボタンがクリックされたときの処理をおこなうことができる。

Fig. 9は、三次元積み付けパターンを表示したウィンドウ上に整数入力のためのウィンドウ（右上）とイメージ環境設定のためのウィンドウ（右下）およびシステム環境設定のためのウィンドウ（左上）を重ね、画面をマルチウィンドウにした状態を示している。

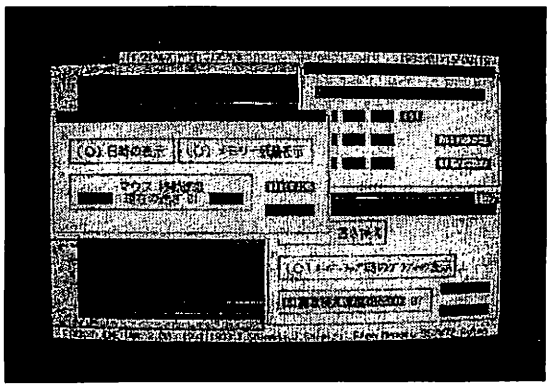


Fig. 9 Multi－Window function

3.4 配置決定システムの構成

マルチウィンドウシステム上でアプリケーションを構築することが可能となる方法を以下に述べる。ハードウェアとしてマウス・キーボードおよびディスプレイを用いて、アイコン・ボタンやウィンドウおよびこれらを複数管理することなどは、マルチウィンドウシステムの基本となる機能である。新しいアプリケーションを構築するには前述したWindowクラスをもと、配置決定に関するクラスを作成することで、容易にマルチウィンドウ機能を追加することが可能となる。

Fig. 10は配置決定システムを構成するクラスの継承関係を示している。各々のクラスの機能は以下のとおりである。

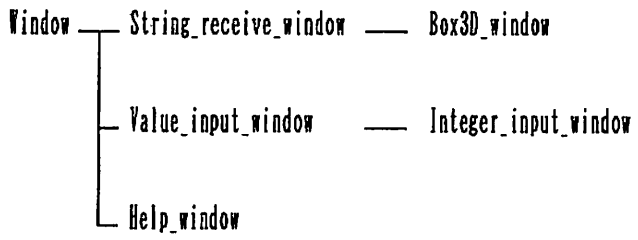


Fig. 10 Sequence of loading pattern determination class

(1) Window クラス
- オーバーラップができるほか画面内で自由にその位置を変えたりサイズを変更できる。
- ウィンドウを単位として配置決定システムを作成している。

(2) String_receive_window クラス
- 文字入力を受け付けるための抽象クラス。
- Windowクラスより継承。

(3) Box3D_window クラス
- 積み付けパターンを三次元立体表示するためのクラス。
- String_receive_windowより外部とのインターフェースだけを継承

(4) Value_input_window クラス
- 数字入力をおこなうためのクラス。
- 入力された数値は String _ receive

windowクラスのオブジェクトに送ることを前提としている。

(5) Integer_input_window クラス

• 整数入力をおこなうためのクラス。

アプリケーションはWindowクラスからの継承で構成すればよい。まず文字入力受け入れのための抽象クラスであるString_receive windowクラスを継承する。積み付けパターンを三次元立体表示するためには、(1) 包装品の縦・横・高さを収得、(2) パレットの縦・横を収得、(3) 積み段数を収得、(4) 積み付けパターンデータのファイルを読み込む、(5) 最適な積み付けパターンを決定等の処理をBox3D_windowクラスでおこなう必要がある。またそれぞれの形状に関する値は整数入力をおこなうためのクラスであるInteger input_windowクラスで再入力を可能とし、再入力が完了した時点で、このウィンドウはString_receive_windowに値をメッセージとして送るようにしている。Box3d_windowがString_receive_windowから継承されているのはこのメッセージを正しく受けとめるためである。

3.5 本システム構築結果と考察

Fig. 11 (1) (2) (3) は包装品の縦、横、高さ、パレットの縦、横および積み段数を入力したときの積み付けパターンの3次元イメージ表示をおこなっているディスプレイ画面である。

Fig. 12は包装品の縦400.0mm、横200.0mm、高さ200.0mm、パレットの縦1100mm、横1100mmおよび積み段数5の積み付けパターンである。このときの積載効率は59.5%である。そこで内容品の梱包方法の変更および形状の変更が可能な製品であれば、包装品の縦を400.0mmから380.0mmに変更することにより、積載効率は84.8%になる(Fig. 13)。さらに包装品の縦を400.0mm、380.0mmから300.0mmに変更することにより積載効率は59.5%、84.8%から99.2%に大きく効率を向上せさることが可能となる

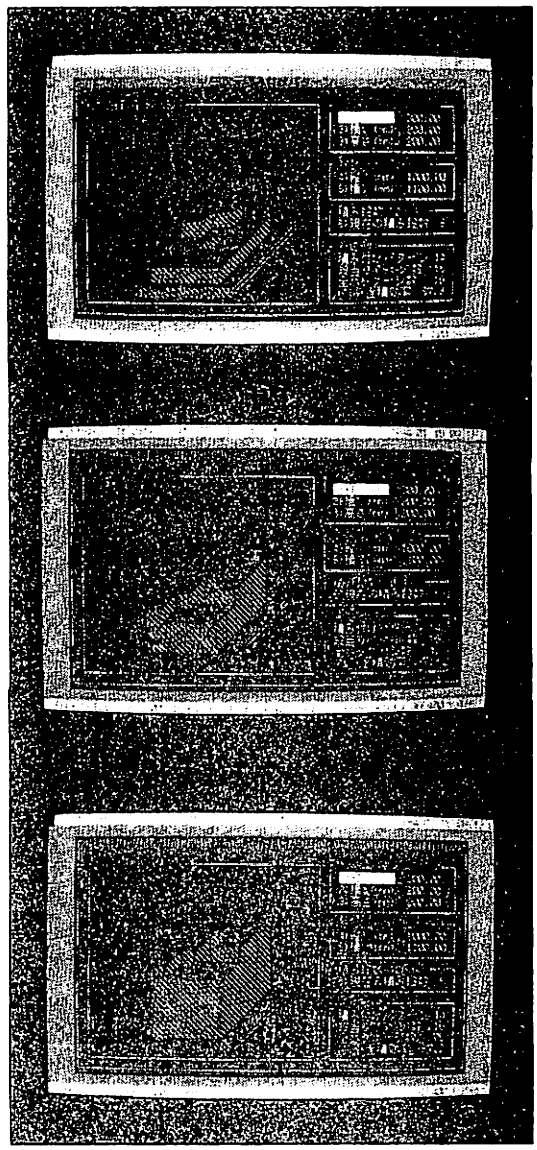


Fig. 11 Loading pattern making

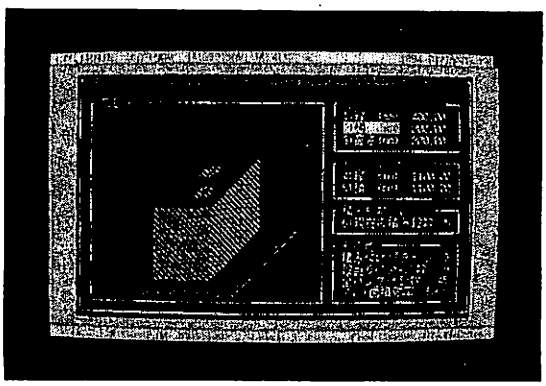


Fig. 12 Object－oriented loading pattern design(1)

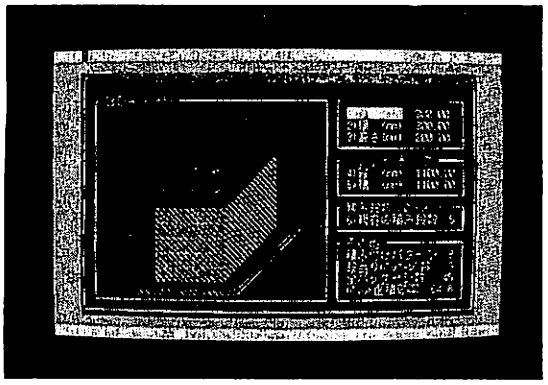


Fig. 13 Object－oriented loading pattern design(2)

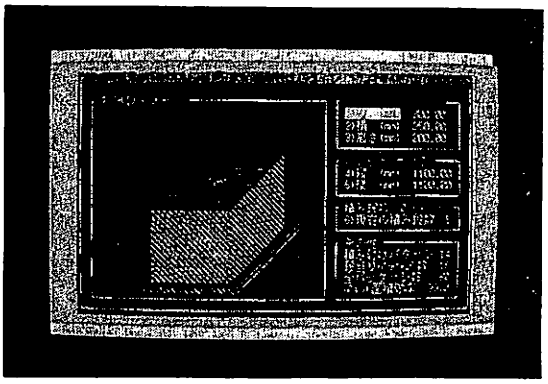


Fig. 14 Object－oriented loading pattern design(3)

(Fig. 14)。このような設計結果について図形と評価値をディスプレイに表示し、最適な値になるように逐次設計作業を繰り返すことで操作性のよいシステムを構築することが可能となる。

また上述したマルチウィンドウ機能を有するシステムの利点は、積み付け作業の仕様変更に対して、特に有効である。なぜならそれぞれの処理が部品化（オブジェクト化）されているため、あらたな業務内容に対応する部品（オブジェクト）を従来の処理につけ加えるだけで、新規ウィンドウを開いて処理することが可能となるためである。

このシステムは包装・物流業務のなかの積み付け作業に的を絞って作成したが今後包装・物流設計の全般に広げていく予定である。

<引用文献>

1) 島田哲夫、山崎潔、一森和之、"パッケージ設計用CADシステム入門"、日報、(1990)
2) Shimada, Yamasaki, The development of a strategic information system applied to packaging design, PACKAGING TECHNOLOGY & SCIENCE, 6 (2) 79－86, John Wiley & Sons
3) 島田哲夫、潮俊光、"戦略情報処理入門"、ダイヤモンド社、(1993)
4) B. ストラウストラップ、プログラミング言語C第2版、トッパン、(1993)
5) スコットマイヤー、EffectiveC、ソフトバンク、(1993)

(原稿受付1994年1月21日)
(審査受理1994年7月22日)