数式処理 *J.JSSAC (2005)*
Vol. 11, No. 3,4, pp. 77 - 88

Noda2005 論文

MathBlackBoard

出口 博章*

神戸大学大学院 国際協力研究科

概　要

MathBlackBoard is a Java program based on the blackboard applet. We can use the blackboard applet with GUI operations. The blackboard applet has simple kernel as computational engine. In this paper, we describe improvements of MathBlackBoard. We developed MathBlackBoard as sets of objects. GUI operations of MathBlackBoard have been improved. And, we added a new button which use certain computer algebra systems as engine.

1　はじめに

MathBlackBoard は，1997 年に松嶋 [1] が作成した Java アプレットである黒板アプレットを，改良・発展させた Java プログラムである．黒板アプレットは簡単な数式処理システムであり，数式の入力と操作，数式処理，そしてグラフの描画が可能となっている [2]．黒板アプレットの一番の特徴は，GUI 中心のユーザインタフェース（図 1）であり，入力装置はマウスなどのポインティングデバイスのみで利用できる．また，簡単な数式処理エンジンを備えているため，Java 実行環境の上で単体で独立して動作する．

MathBlackBoard の開発においては，黒板アプレットに対して，GUI 操作の細かな変更や新たな操作の追加，また，外部の数式処理エンジンに接続するための機能の追加などを行なった．MathBlackBoard によって目指すものは，数式を入力・編集・操作・処理するための使いやすいユーザインタフェースの実現である．

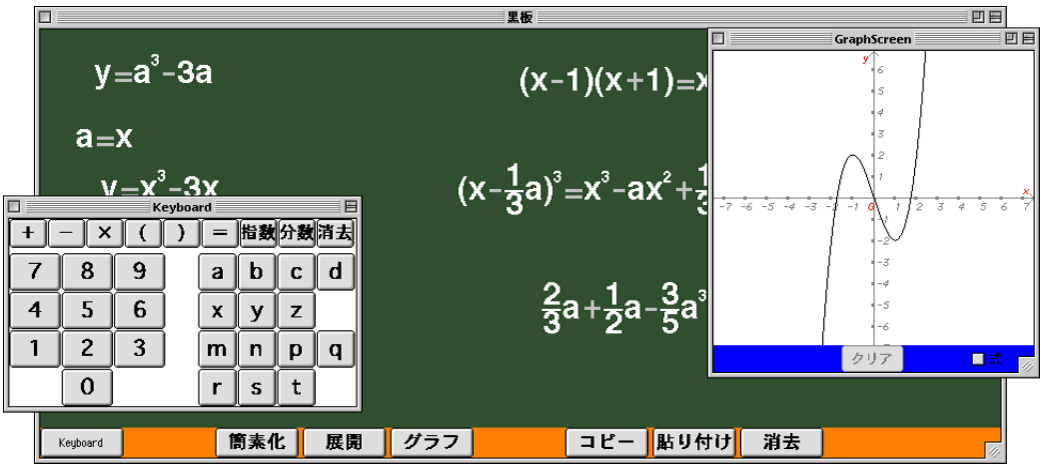
2　黒板アプレット

松嶋は 1997 年当時の数式処理システムの問題点を「複雑な CUI[1)] 入力」と「OS 依存」であると考え，その問題点の解決の糸口として「わかりやすい」「手軽である」の二点を挙げ，「使いやすい数式処理システム」のサンプルとして黒板アプレットを提示した．以下に「わかりやすい」「手軽である」についての説明と，その実現のための手段を，文献 [1] に基づいてまとめたものを示す．

*deg@kobe-u.ac.jp

1) Character User Interface： 文字入力中心のユーザインタフェース．

図 1: 黒板アプレット



分かりやすい： 『利用者が即座に正しい操作方法を推測しやすく作られていること』（**実現手段：**数式処理のボタン化や，マウス操作による数式への代入などの GUI 強化）

手軽である： 『システムの更新が容易であることと，すぐに使える環境にあること』（**実現手段：**Web ブラウザの利用，Java アプレットの利用）

2.1 分かりやすさの実現

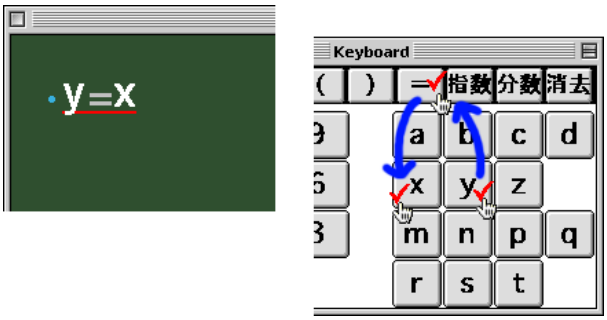
黒板アプレットにおいて強化された GUI 操作を以下に挙げる．

数式の入力

- GUI ボタンによる数式入力
  黒板アプレット上に表示される数式は，キーボードウィンドウに並んだ GUI ボタンのクリックで入力する．全ての入力をキーボードウィンドウのボタンで行なうため，入力装置はマウスなどのポインティングデバイスのみで良い．
  図 2 は，キーボードウィンドウのボタンによって，黒板ウィンドウに $y = x$ が入力される様子を示している．まず，キーボードウィンドウの「y」ボタンをクリックする．次に「=」ボタンをクリックする．そして最後に「x」ボタンをクリックする．これらの一連の操作によって，黒板ウィンドウに $y = x$ が表示される．
- 任意の場所に入力可能
  それぞれの数式は表示される位置の座標情報を持っており，テキストエディタのように行の位置に縛られずに表示される．
  黒板ウィンドウ上の任意の場所をクリックすると，その位置に入力位置を示すカーソルが

表示される．例えば図 2 の黒板ウィンドウ上の $y = x$ の場合は，y の左に表示されている小さな円がカーソルである．このカーソルはドット単位で位置が指定され，入力される数式の基点となる．

図 2: キーボードウィンドウのボタン・クリックによる入力



数式の処理・操作

- GUI ボタンによる数式の処理（展開，簡素化，グラフ描画）
  数式の処理は，数式をクリックして選択後に，処理に応じたボタンをクリックすることによって実行される．
  例えば既に入力されている $(x+y)^2$ の展開を行なう場合は，`Expand[]` のようなコマンド入力を利用するのではなく次のような操作を行なう．まず，$(x+y)^2$ をクリックして選択する．すると $(x+y)^2$ に選択されたことを示す下線が表示される．次に黒板ウィンドウの下に並んだボタンのうち「展開」ボタンをクリックする．すると元の式の右に $= x^2+2xy+y^2$ が連結され，表示は $(x+y)^2 = x^2+2xy+y^2$ と変化する．
- ドラッグ＆ドロップによる数式の操作（連結や代入）
  入力された数式が複数ある場合の数式どうしのドラッグ＆ドロップには，操作対象となった数式どうしについて，連結や代入の処理が割り当てられている．
  図 3 は，$a = x-1$ を代入のための情報として利用して，$y = a^3-3a$ の a に $x-1$ を代入するための操作を示している．$a = x-1$ と $y = a^3-3a$ は既に入力されているものとして，まず $a = x-1$ をドラッグする（図 3 左）．この時，図 3 左にあるように，$a = x-1$ の複製がマウスカーソルと共に移動する．次に，そのまま $a = x-1$ を $y = a^3-3a$ の上までドラッグしてドロップする（図 3 中央）．ドロップ直後に，処理方法の選択肢が表示される．これらのうち「代入」を選択すると，$y = a^3-3a$ の表示が $y = (x-1)^3-3(x-1)$ となる（図 3 右）．連結の場合は，ドロップ直後の選択肢で「左辺」「右辺」または「両辺」のいずれかを選択する．図 3 で連結を選択した場合は，それぞれ a，$x-1$，または $a = x-1$ が $y = a^3-3a$ の右に連結される．

図 3: ドラッグ＆ドロップによる代入（左：ドラッグ, 中：ドロップ, 右：実行後）

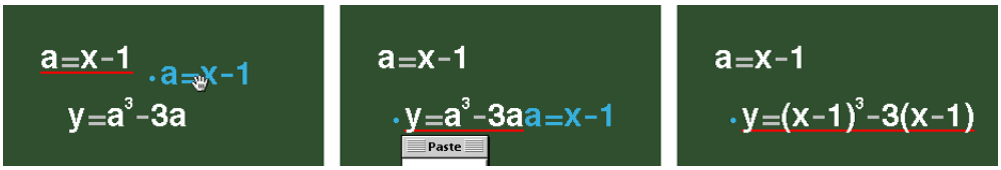


表 1: 比較

	「数式」の表現	「数式の操作」の表現
コマンド入力	文字列 （例：(x-y)^2 ）	文字列 （例：Expand[]）
黒板アプレット	文字列 （例： $(x-y)^2$ ）	GUI 操作 （例：「展開」ボタンのクリック）

以上のように GUI 操作を利用することによって，数式は黒板上に表示されている文字列で表現され，数式の処理は GUI 操作で表現されるというように，表現が明確に区別される（表 1）．表 1 の関係に似た例として，OS におけるファイルの表現とコピー操作の表現についての，コマンド入力に基づく OS と，GUI に基づく OS との比較が挙げられる．OS におけるファイルのコピー操作は，コマンド入力ではファイルは文字列（ファイル名），操作も文字列（cp や copy）と同じように表現され，一方，GUI ではファイルは GUI パーツ（アイコンとファイル名），操作は GUI 操作（ドラッグ＆ドロップ）と明確な区別のある表現となっている．

黒板アプレットでは，数式そのものの表現と数式の操作の表現に，GUI 上の別の表現を用いることによって，表示画面を実際の黒板やノートに近付けることができた．また，操作時にはポインティングデバイスを利用して実際に手を動かすという，直感的なユーザインタフェースが提供されている．以上が黒板アプレットにおける「分かりやすさ」である．

2.2 手軽さの実現

黒板アプレットは web ブラウザから呼び出される Java アプレットとして作成されている．そのため，Java 実行環境を呼び出せる web ブラウザと Java 実行環境がインストールされていれば黒板アプレットが利用可能であり，Windows や Mac OS，UNIX 系の OS など多くの異なるシステム上で同じ Java アプレットが共用できる．

また，Java アプレットとして利用すれば更新はサーバ側で行なうだけで良く，メンテナンスの負荷が軽減する．新たな黒板アプレットの利用には，利用時に再度ダウンロードを行なえば良く，一般的なソフトウェアのインストール作業よりも負荷は軽くなる．

これらが黒板アプレットにおける「手軽さ」であった．ただしこれらは，1997 年当時では特徴と言えたが，サーバサイドの Java も発展した現在では大きな特徴とまでは言えない．

図 4: Palm OS 版でグラフ描画（左：入力途中, 中：グラフボタンを押す, 右：実行後）

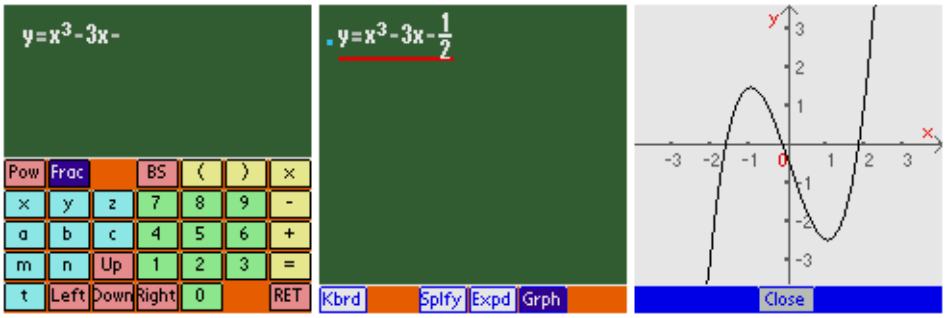
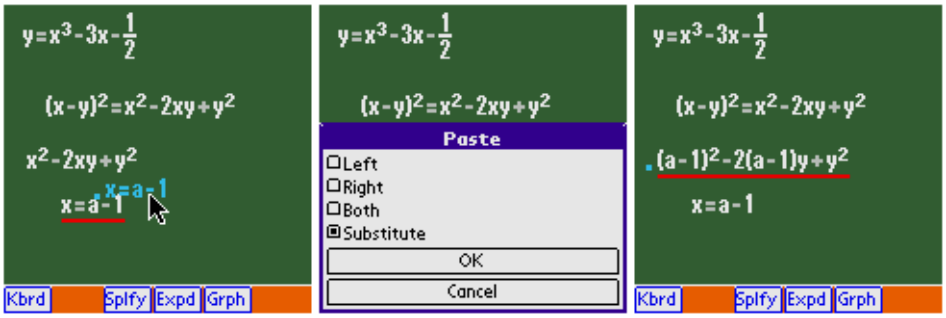


図 5: Palm OS 版でドラッグ&ドロップ（左：ドラッグ, 中：代入を選択, 右：実行後）



ところで，黒板アプレットは JDK 1.1[2] の Java 実行環境であっても動作する状態を維持して開発が続けられたため，当時では考えられなかった PDA での利用も可能となった．PDA のハードウェア仕様はパソコンより劣るため，PDA の Java 実行環境はパソコン用のものよりも機能の少ない仕様とならざるを得ない．しかし，JDK 1.1 でも動く黒板アプレットは MIDP[3] を利用した Palm OS 端末や，PersonalJava[4] を備えた Zaurus などでも利用可能となっている（図 4，図 5，図 6）．これらは Java アプレットとして web ブラウザから呼ばれるものではないが，それぞれの環境に合わせた Java 実行環境によって実行されている．

また黒板アプレットは，前述のようにマウス操作のみで利用可能であるため，ペン操作中心の PDA においても違和感なく使用することができる．単に Java アプレットであったということのみでなく，ポインティングデバイス中心に設計されていたということも合わせて考えると，黒板アプレットが「手軽さ」を備えていることは確かである．

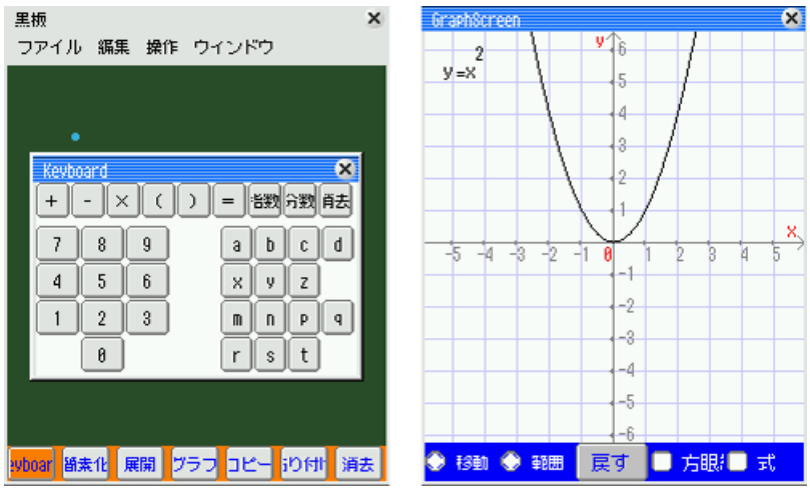
---

[2] Java Platform 1.1 API and Documentation：`http://java.sun.com/products/archive/jdk/1.1/`

[3] MIDP for Palm OS： `http://java.sun.com/products/midp4palm/`

[4] PersonalJava Application Environment：`http://java.sun.com/products/personaljava/`

図 6: Zaurus 版（左：起動直後, 右：グラフ描画）



3 MathBlackBoard

3.1 黒板アプレットの改良の余地

ここまでに述べてきたように，黒板アプレットは優れたユーザインタフェースを備えたものであったが，細かい点では改良の余地も残されていた．いくつか以下に挙げる．

- 数式のドラッグ＆ドロップによる単純な「移動」ができない
- 数式の数式処理ボタンへのドラッグ＆ドロップに対応していない
- メニューバーを備えていない
- マウスの右ボタンクリックに対応していない
- 描画されたグラフの操作ができない
- 備えている数式処理エンジンは多くの数式処理には対応していない

黒板アプレットのオリジナル版では，数式をドラッグした後に，黒板ウィンドウ内の黒板エリア（黒板のように緑色をした部分）上にドロップした場合は，「コピーした項目をペーストするためのダイアログ」が表示される．Palm OS 版を示した図 5 の中央に同様のダイアログが表示されているが，利用者はこの中から「左辺」「右辺」「両辺」または「代入」を選択し「OK」を押すか「キャンセル」を押すことになる．

ここで取り上げている「黒板エリアへのドラッグ＆ドロップ」の場合では，「代入」を選択した場合と「キャンセル」を選択した場合には数式に変化はなく，カーソルのみが移動する．それ以外のものを選択した場合には，選択に応じてドロップした位置に数式が表示されるが，ドラッグされた元の数式は残ったままである．つまり，「黒板エリアへのドラッグ＆ドロップ」は数式のコピー＆ペーストの意味を持っており，OS におけるドラッグ＆ドロップとは違う意味を持つ操作となる．利用者にとってのハードルを低くするためには，GUI 操作に OS と同等または類似

の意味を割り当てるのが望ましいため，この点においては改良の余地があった．

また，一般的な利用者が多様な GUI 操作に慣れ親しんでいる昨今では，数式の数式処理ボタンへのドラッグや，メニュー操作，あるいはマウスの右ボタンクリックなど，GUI パーツや GUI 操作に何らかの意味を割り当てることが望まれた．さらに，描画されたグラフに対しての GUI 操作や，グラフの再描画，グラフの比較など，GUI 操作の面では様々な拡張の余地があった．

利用する数式処理エンジンについては，J/Link[5] による Mathematica や，OpenXM[6] による Risa/Asir など，外部からの利用に対応した数式処理システムの仕組みが整ってきたこともあり，既にある数式処理システムを数式処理エンジンとして利用できるように改良することが求められていた．また，数式処理エンジンを統一的に web から扱うために考案した Middle-regulator[3] の開発において目指したのは，裏で動いている数式処理システムをユーザに意識させないために，ユーザインタフェースと数式処理システムの中間に位置する調節役であった．Middle-regulator それ自体は，サーバ・サイドの技術進歩により陳腐化したため，永らく眠っている状態であるが，その目指した方向性を受け継ぎ，複数の数式処理エンジンを扱う事も黒板アプレット改良の視野に入れた．

3.2 MathBlackBoard における改良点

黒板アプレットの開発過程で Java アプレットとしてのみでなく Java アプリケーションとしても起動可能とした．そして，制限のより少ない Java アプリケーションとしての利用を主な利用方法として位置付け，改良を進めた．名称は二転三転した結果，黒板アプレットから MathBlackBoard へと改めた．以下に主な改良点を挙げる．

3.2.1 黒板エリアへのドラッグ＆ドロップ

MathBlackBoard では，黒板エリアへの数式のドラッグ＆ドロップは，ダイアログを出さずに移動するよう変更した．この操作は OS の GUI 操作においてはごく一般的であるため，黒板上に表示される数式の扱いに対してのハードルは低くなったと言える．

例えば $y = x - 3a$ と $y = x^2$ が上下に並んで表示されている時に，$y = x - 3a$ をドラッグして $y = x^2$ の右にドロップすれば，$y = x^2$ の上にあった $y = x - 3a$ が $y = x^2$ の右に移動する．その際，ダイアログは表示されず，すぐに表示が更新される．

3.2.2 数式処理ボタンへのドラッグ＆ドロップ

また，数式処理ボタンに数式をドラッグ＆ドロップした場合には，「処理される対象」を「処理の実行（の象徴）」へドラッグ＆ドロップするという意味になり，OS の備える GUI 操作と同等の意味であるため利用者にとっては直感的に理解しやすい操作となる．

OS においては，書類をアプリケーションのアイコンにドラッグ＆ドロップすることによって「ドラッグされた書類をドロップされたアプリケーションで開く」という操作を行なうが，MathBlackBoard では，例えば $2x^2 - 3 - x^2$ をドラッグして「簡素化」ボタン上にドロップする場合には「ドラッグされた数式 $2x^2 - 3 - x^2$ をドロップされたボタンの処理内容（簡素化）で処

[5] Java Toolkit: J/Link： http://www.wolfram.com/solutions/mathlink/jlink/

[6] OpenXM (Open message eXchange for Mathematics)：http://www.openxm.org/

理する」という意味の操作となり，$2x^2 - 3 - x^2$ は $x^2 - 3$ と変更される．

3.2.3 GUI メニュー

そして MathBlackBoard には，GUI 操作を基本とする OS では一般的であるメニューバーとポップアップメニューも追加された．数式処理ボタンが配置できる場所には限りがあるため，GUI メニューの利用によって多様な操作を提供することが可能となった．

メニューを利用して処理を行なう場合には，例えば，$(x+y+z)^3$ を選択して下線が表示されている状態で，メニューの「展開」を選択するというようにして行なう．

3.2.4 グラフウィンドウのモード

一方，グラフに関しては，まず「移動モード」と「範囲選択モード」を用意し，その二つを切り替えるための GUI ラジオボタンを配置した．「移動モード」とはグラフを上下左右に動かすモードで，グラフウィンドウ内に表示されている範囲がマウスなどのドラッグ操作により変化する．また，「範囲選択モード」はマウスなどのドラッグ操作によって指定された矩形をグラフウィンドウの範囲まで拡大させるモードで，同じ操作にキーボード操作を併用することで縮小させることも可能である．それらの操作によって変化が加えられたグラフを元に戻すために「戻す」ボタンも配置されている．さらに装飾的なものとして，方眼線や数式をグラフウィンドウ内に表示させるための GUI ボタンが配置されている．

図 6 右は Zaurus 版黒板アプレットのグラフウィンドウだが，グラフに関する改良を行なった後に Zaurus で実行されたものである．この図では方眼線と数式の表示状態がオンになっている．図 4 右には，Palm OS 版で実行された，改良前のグラフウィンドウが表示されている．

3.2.5 グラフウィンドウへのドラッグ＆ドロップ

また，グラフウィンドウに対する数式のドラッグ＆ドロップによって数式の追加や再描画が行なわれるようになった．既にグラフが描画されているグラフウィンドウに，グラフ描画の元となった数式とは別の数式をドラッグ＆ドロップすることで，複数の数式を同時に表示させることが可能である．グラフの再描画は，描画の元になった数式に変更を加え，変更後にグラフウィンドウにドラッグ＆ドロップすることによって実行される．これらの操作によって，複数のグラフを比較したり，数式の変化がグラフに与える影響を調べるといった学習ソフトとして利用する際のツールとしての機能が提供されている．

例えば，$y = x^2$ と $y = (x-1)^2$ のグラフを同時に表示させる場合には次のように操作する．まず $y = x^2$ をキーボードウィンドウを利用して入力する．次に $y = (x-1)^2$ を別の場所に入力する．この時，最初の $y = x^2$ をコピー＆ペーストして編集することによる入力も可能である．そして，$y = x^2$ を選択して「グラフ」ボタンをクリックすることによって $y = x^2$ のグラフが表示されたグラフウィンドウを出す．最後に $y = (x-1)^2$ を $y = x^2$ が表示されたグラフウィンドウまでドラッグして，そのグラフウィンドウ上でドロップする．

また例えば，グラフ $y = x^2$ と $y = (x-1)^2$ が表示されているグラフウィンドウのグラフ $y = x^2$ を $y = x^2+7$ のグラフに変更する場合の操作は次のようになる．まず，黒板ウィンドウ上の $y = x^2$ を編集して $y = x^2 + 7$ と変更する．この段階ではグラフ表示に変化はない．そして $y = x^2 + 7$

を，グラフ $y = x^2$ と $y = (x-1)^2$ の表示されたグラフウィンドウまでドラッグさせ，そのグラフウィンドウ上でドロップする．すると，グラフウィンドウには $y = x^2 + 7$ と $y = (x-1)^2$ のグラフが表示された状態となる．

4 MathBlackBoard の実装

MathBlackBoard はオブジェクトの集まりとして再構成され，オブジェクトの独立性が高まった．その結果，機能の追加が容易になり，保存・読込み機能，取り消し機能，そして外部の数式処理エンジン利用のための機能などが追加された．

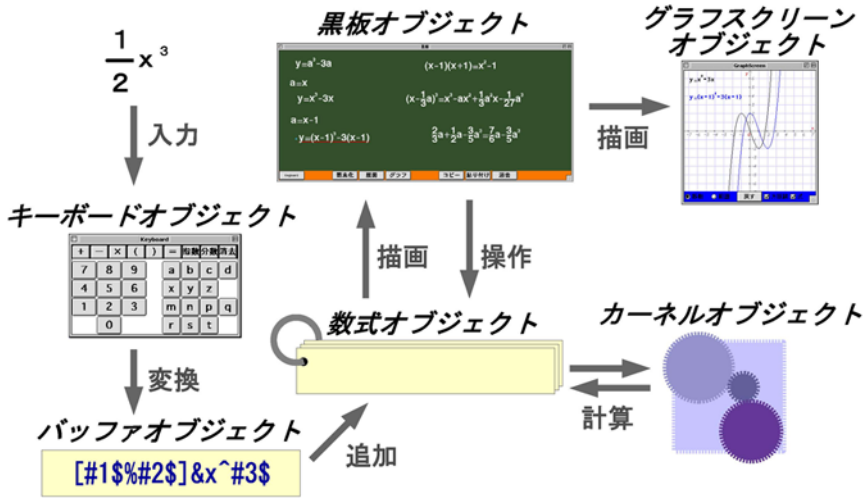
4.1 MathBlackBoard を構成するオブジェクト

MathBlackBoard は大きく分けて 6 種類のオブジェクトで構成されている．それらの関係を図 7 に示す．

黒板オブジェクトは数式の入力・出力や数式操作のための場所となるオブジェクトである．キーボードオブジェクトには GUI ボタンが並び，数式入力の際に利用される．グラフスクリーンオブジェクトでは数式のグラフ描画が行なわれる．また，数式はそれぞれが数式オブジェクトとして生成される．以上は GUI パーツとしてコンピュータディスプレイ上に表示されるオブジェクトである．

GUI パーツとして表示されないオブジェクトには，数式編集などに利用されるバッファオブジェクトと，数式を処理して結果の数式を返すカーネルオブジェクトがある．

図 7: MathBlackBoard を構成する主なオブジェクトの関係



これまで述べてきた黒板アプレットから MathBlackBoard への改良に伴って，オブジェクトが整理され，以上のように 6 種類に大別できるようになった．そして，オブジェクトの機能と境界が明確になり，オブジェクト間のインタフェースが整ったことによって，さらに機能の追加が容易になった．

4.2 保存・読込み機能と取消し機能

数式オブジェクトと他のオブジェクトとのインタフェースが整い，オブジェクトの境界が明確になったため，数式オブジェクトを取り出して扱う事が可能となった．そこで，ファイル保存するために Java の Serializable インタフェースを，数式オブジェクトに実装し，保存と読込みを可能とした．図 7 にリングで束ねられたカードで表しているように，複数の数式オブジェクトが一つの管理用オブジェクトに束ねられているため，数式の束としてまとめて保存・読込みを行なっている．管理用オブジェクトは表示されている全ての数式オブジェクトを参照しているため，管理用オブジェクト一つを保存することによって，その過程で参照されている全ての数式オブジェクトも同時に保存される．

取消しについても保存・読込みと同様に，参照された数式オブジェクト全てと管理用オブジェクトをまとめて複製して内部に保持すれば実装できるが，PDA で利用することも視野に入れると内部メモリは節約して使いたい．そこで，取消し機能については，数式オブジェクトを束で扱うのではなく，変更された数式オブジェクトのみを記憶しておくことで実現させた．実際には，数式を束で管理している小さなオブジェクトのコピーをメモリに保存して，取り消しの際に呼び出すようになっている．図 7 ではリングで表現されたこの管理用オブジェクトは，数式オブジェクトに変化がない場合は，古いオブジェクトをそのまま参照し，変化があったオブジェクトへの参照だけを入れ替える．そのため，保存された管理用オブジェクトは以前のオブジェクトへの参照を保持しているので，結果として変更される前のオブジェクトを呼び出す事ができる．どこからも参照されなくなったオブジェクトはガーベッジコレクタによって消去されてしまうが，参照されている限りは消されることはないため，復元可能である．

4.3 MathBlackBoard の数式処理オブジェクト

黒板アプレットにおいては，カーネルオブジェクトは複数のオブジェクトで構成されていたが，MathBlackBoard において一つのカーネルオブジェクトとして統合させた．統合の際にオブジェクト間のインタフェースを整え，境界が明確になったため抽象化が可能となった．そこで，スーパークラスとしてカーネル抽象クラスを定義し，従来のカーネルオブジェクトをそのサブクラスとして定義しなおした．従来のカーネルオブジェクトは MathBlackBoard のデフォルトカーネルとすると共に，外部の数式処理エンジンを利用するために Mathematica や webMathematica, OpenXM と通信を行なうカーネルサブクラスをそれぞれ実装した．

MathBlackBoard における数式処理はカーネル抽象クラスで定義されたメソッドを利用するよう変更されたため，サブクラスの切り替えによって利用される数式処理エンジンの変更が可能となった．

Mathematica には J/Link を利用して接続し，webMathematica には HTTP を利用して接続している．また，OpenXM に接続するには OpenXM のサイト内[7] へアクセスして以下の手順で JAR ファイルを作成する．以下は Windows のコマンドプロンプトでの操作を想定している．

- Source Distribution (Download) から `openxm-head.tar.gz` をダウンロードする．

[7] OpenXM HEAD： `http://www.math.sci.kobe-u.ac.jp/OpenXM/1.2.2/index.html`

- 展開してできた OpenXM フォルダの，OpenXM\src\OpenMath まで移動する．
- MathBlackBoard では JDK 1.1 をターゲットとしているため，

  ```
  >javac -target 1.1 ORG\openxm\tam\OpenXM.java
  >del ORG\openxm\tam\*.java
  >jar cvf OpenXM.jar ORG
  ```

  と実行して OpenXM.jar を作成する．
- 適当な場所に置き，システムの環境変数 CLASSPATH で OpenXM.jar を指定する．
  J/Link 利用のために JLink.jar も指定しているため，例えばこのようになる．

  ```
  C:\Program Files\...（省略）...\JLink.jar;OpenXM.jar;.
  ```

  このように指定しておくと，実行時のパスに OpenXM.jar があれば呼び出される．

OpenXM.jar 作成後の使い方は OpenXM\src\OpenMath\testclient.java を参考にした．現状ではサーバとして OX_ASIR を立ち上げておき，OX_ASIR 起動時に設定したポートに対してアクセスする必要がある．

以上のように外部の数式処理エンジンの利用が可能になり，黒板アプレットが元から備えていたデフォルトカーネルでは処理できなかった因数分解も処理できるようになった．

抽象クラスによって，呼び出し側からはどの数式処理エンジンを利用しているのかを意識せずに利用可能となり，サーバ・サイドかクライアント・サイドかの違いはあるものの，かつて Middle-regulator[3] の開発によって目指したものも実現されている．

5 考察

黒板アプレットから MathBlackBoard へと開発を進めていく中で，オブジェクトの見直しと統合が行なわれ，結果としてオブジェクトの独立性が高まった．そして次の段階として，そのオブジェクトの独立性に支えられて，保存・読込み機能や取消し機能，外部カーネル利用機能など開発が進められた．しかし，外部の数式処理エンジンを利用可能とはなっても，現状ではその能力のほんの一部を拝借してるに過ぎない．

5.1 今後の課題

そこで，今後の開発の方針の一つとして，外部の数式処理エンジンの能力をさらに広い範囲で利用するという方向性がまず挙げられる．この方向性では，まず，今回追加された因数分解処理以外にも数式処理を追加することが挙げられるが，それ以外にも課題はある．それぞれの外部数式処理エンジンの備えるグラフィックス出力をいかに統一的に取り込むかという問題を解決するために，グラフウィンドウに関する開発が必要である．単にグラフィックスのデータを取り込んで表示させるだけではなく，インタラクティブなグラフ操作が可能となるような枠組みを作成しなければならない．

また，MathBlackBoard の問題点として，表示できる記号がまだ少ないという事と，数式の内部表現が拡張性の高いものではない事が挙げられる．よって，二つ目の開発方針は数式の内部表現の見直しと，それに伴って増えるであろうと考えられる，表示できる記号の分かりやすい GUI パーツ配置の検討という事になる．

他方, ユーザインタフェースのハードルを低くすることの目的の一つは, コンピュータリテラシの修得度合いが様々であろうと思われる利用者を想定した, 教育における利用である. 教育利用の方向性での開発方針では, いわゆる電卓的な使い方を提供することは考えていないため, 数式処理機能は表示の判定や検算などの補助的な役割で利用されることが想定される. MathBlackBoard が教育現場で利用されるイメージとして現段階で思い描いている, あるべき姿は「何らかの数学問題を解く際に, 紙と計算用紙のセットと MathBlackBoard のどちらか好きなほうを使って良いなら MathBlackBoard を使いたくなる」ようなツールである. 入力したものをパッと計算して答えを出してくれるような電卓の代替ではなく, 頭の中で行なわれる計算や思考の手助けのための計算用紙の代替となるものが, この方向性で目指すべきゴールではないかと考えている. このゴールは, 上に挙げたような他の開発方針を進めながらも, 遠いゴールとして常に意識しておくべき性質のものである.

参 考 文 献

[1] 松嶋純也: Java を用いた使いやすい数式処理システム, 神戸大学大学院教育学研究科 修士論文, 1998.

[2] 出口博章: 黒板アプレット, 数式処理 *Vol9 No.1*, pp.32-37, 2002.

[3] DEGUCHI Hiroki: The Integrated Use of Computer Algebra Systems across the Internet, *Proceedings of the 1998 Asian Symposium on Computer Mathematics*, pp.101-105, 1998.