Anchor Garden: オブジェクト指向言語における データ構造の基本を理解するためのワークベンチ

三浦元喜† 杉原太郎† 國藤 進†

我々は Java や C# などの静的な型付けを行うオブジェクト指向言語を学習するうえでつまずきやすい「型・変数・オブジェクトとデータ参照」の理解を促進するためのワークベンチ Anchor Garden を提案する．学習者はマウス操作により視覚的な表現をもつ "変数" や "オブジェクト" を生成したり，変数からオブジェクトにリンクを張る操作を行いながら，プログラミングにおける独特の考え方や概念世界に接近することができる．また本ワークベンチは概念世界における操作に対応したコードを自動生成する機能を持つ．これにより学習者は生成されたコードと自身が行ったモデルの操作を対応付けやすくなる．実験の結果，Anchor Garden に習熟すると変数とオブジェクトの関係の理解能力が向上する傾向が見られた．また変数とオブジェクトの関係理解度とプログラム作成能力との相関がみられ，Anchor Garden がプログラム作成能力を向上する可能性が示された．

Anchor Garden: A Workbench for Understanding Fundamentals of Data Structure in Object Oriented Programming Language

Motoki Miura,† Taro Sugihara† and Susumu Kunifuji†

We propose "Anchor Garden," a workbench software to learn fundamentals of data structure with concepts of type, variable, object, and its relations in a strongly-typed object oriented programming language such as Java and C#. Learners can approach the concepts by a direct manipulation of graphical models. Anchor Garden (AG) allows the learners to create variables and objects, and to link among them. Since AG automatically generates source-code corresponding to the learner's manipulations, the learner can relate manipulations and representations of source code. Experimental result showed a positive tendency of learning effect with AG, and high correlations between the concept understanding and programming ability. Thus AG has a possibility to enhance the programming ability for novice programmers.

1. はじめに

これまでプログラミング学習者の理解を促進するためのさまざまな環境が構築・運用されてきた．ソースコードの動作をアニメーションによって表現する Jeliot3[1] や，3次元キャラクタにメッセージを送り，動作させながらプログラミングの基礎を学ぶ Alice[2]，オブジェクト指向の考え方に慣れさせることを目的とした BlueJ[3]，アルゴリズムアニメーションを簡易に作成できる JAWAA[4] などである．いずれもプログラミングにおける抽象的な概念をわかりやすく伝えるため，グラフィカルな表現やアニメーションを利用して理解を促すことを目的としている．しかし既存のシステムの多くは，正しいソースコードを入力しないと期待する動作が得られなかったり，特殊なスクリプト言語を用いており実用的な言語の学習に直接結びつかないといった問題があった．

我々は学習者が実践的なプログラミング言語（特に Java や C#などの静的な型付けを行うオブジェクト指向言語）を修得するうえでつまずきやすい概念である「型・変数・オブジェクトとデータ参照」に対象を絞り，これらの理解を促進するためのワークベンチ Anchor Garden を構築した．本稿における「ワークベンチ」とは，概念理解を促すためのモデルの視覚化に加えて，モデルを直接操作できる環境のことを指す．学習者はワークベンチにおけるモデルとの対話を通じて，どんな操作が可能であるかを探究したり，操作によって概念世界のモデルがどう変化するかを模索したりしながら体験的に学ぶことが可能となる．

† 北陸先端科学技術大学院大学 知識科学研究科
School of Knowledge Science, Japan Advanced Institute of Science and Technology

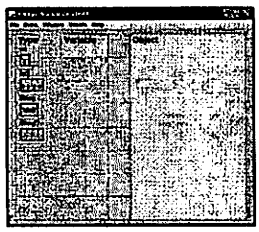

図1 起動画面
Fig. 1 Initial Screen.

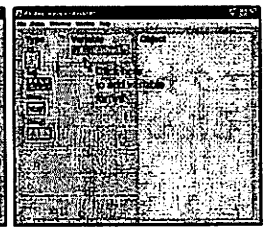

図2 変数生成
Fig. 2 Create Variable.

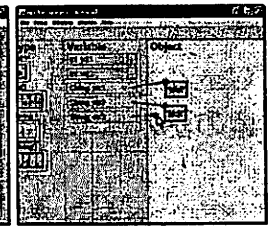

図3 オブジェクトにリンク
Fig. 3 Link to Object.

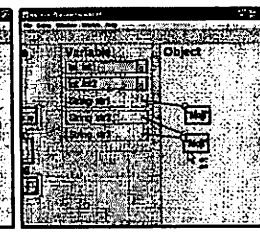

図4 リンク完了
Fig. 4 Linked.

2. Anchor Garden

Anchor Garden（以降，AG）は，静的な型付けを行うオブジェクト指向プログラミング言語における「型」と「変数」および「オブジェクト」を視覚化し，かつ直接操作させることによって，学習者の概念理解を促すことを目的としたソフトウェアである．本システムが対象とする利用者は，変数や計算手順の概念は理解しているが，配列をはじめとするデータ構造や上記言語における具体的なデータ処理方法を修得していない学習者である．従来のアルゴリズムアニメーション環境が，学習者が記述したコードをもとに動作を確認するというアプローチであるのに対し，AG ではコード記述前の段階における，概念形成支援を対象としている．

2.1 基本操作

AG の起動時の画面を図 1 に示す．画面は左から [Type][Variable][Object] の 3 つの領域に分かれており，それぞれ「型」「変数」「オブジェクト」を置くスペースとなっている．[Type] 領域には，int や String など「データ型選択ボタン」が配置されている．学習者はいずれかの「データ型選択ボタン」を押し「データ型」を選択した状態で，[Variable] や [Object] 領域を左クリックすると，選択されたデータ型の変数やオブジェクトが生成され画面に表れる．

int や char といった，Java におけるプリミティブなデータ型を選択した場合，[Variable] 領域で左クリックすると変数と値が生成される（図 2）が，[Object] 領域でクリックしても何も生成されない．String や配列といったコンストラクタによりオブジェクト生成を行うデータ型を選択した場合には [Object] 領域でオブジェクトを生成することができる．学習者は，紙にスタンプを押すような感覚でデータ型を選び，オブジェクトを生成することができる．

学習者は [Object] 領域に手軽にオブジェクトを生成できるが，生成したオブジェクトを放置しておくと，だんだん表示が薄くなり（透明度が高くなり）約 10 秒で完全に消滅する．この表現により，オブジェクトは変数から参照される必要があることを示している．そこで学習者は消滅を避けるため，「変数」から「オブジェクト」にリンクを張る必要がある．オブジェクトにリンクを張るには，変数左の円形のつまみ（リンクタブ）をマウスでドラッグ（図 3）し，接続したいオブジェクト上でボタンを離す．このような「つまんでくっつける」という直感的な操作によって，学習者は変数から参照するオブジェクトを簡単に指定することができる．

このような「変数」と「オブジェクト」を独立して生成し，かつリンクを張る操作を行うことにより，学習者は以下の内容を体験的に学ぶことができる．

- 型と変数，オブジェクト（データ）の関係．
- プリミティブ型（int, char）と，オブジェクト型（String, 配列）の違い．
- オブジェクトを継続的に利用可能にするためには，少なくとも 1 つの変数からリンクで辿れる（参照している）必要があること．
- 1 つのオブジェクト型変数から参照可能なオブジェクトは 1 つであること．
- 1 つのオブジェクトが複数の変数から参照されうること（図 4）．またその際は，データの実体は共有されており，1 つのみであること．（図 4 のマウスカーソル付近にある [str2/str3] という表示は，このオブジェクトを参照する変数やプログラム表記を示している）
- リンクは張り替えたり，外したりすることが可能であること．（リンクをすべて外されたオブジェクトは，急速に表示が薄くなり約 5 秒で消滅する．）
- リンクを外されたオブジェクトの領域は，ガベージコレクション機能によって回収されること．

図 1〜図 4 で示した画面は，初級モードであり，選択できる型が int, String およびその配列に限られている．初級モードで学習可能な内容をすべて理解した学習者は，上級モードに移行できる．上級モード

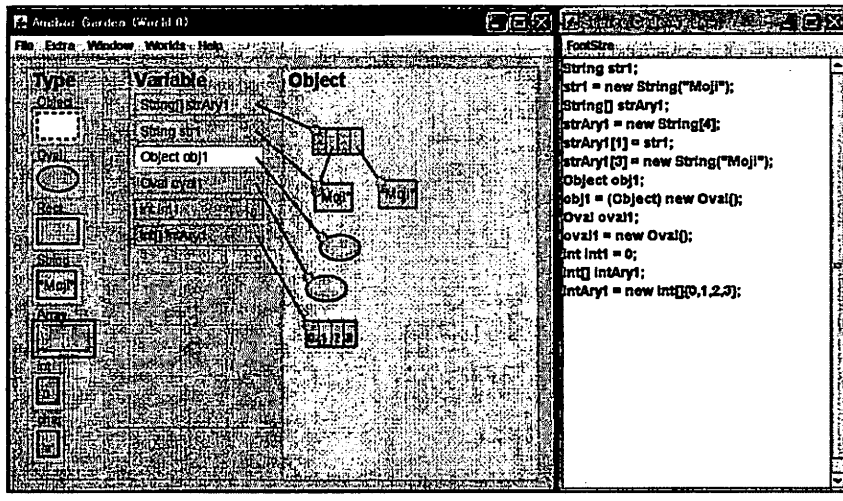

図 5 上級者モードとソースコード
Fig. 5 Advanced mode and source code window.

(図 5) では，基底クラス (Object) や，その派生クラス (Oval,Rect,String) を [Type] 領域に配置し，それぞれのインスタンスを生成することができる．たとえば Object クラスの変数 obj1 から，Oval クラスのインスタンスには参照可能だが，Oval クラスの変数 oval1 から，Object クラスのインスタンスには参照できないといった，クラスの継承関係と参照ルールも理解できる☆．型が異なるなどの理由により，変数からオブジェクトを参照できない場合には，リンクタブを離したときに警告音が鳴り，リンクタブが操作前の状態に戻る．

2.2 ソースコード表示による操作とコードの対応付け

AG では，学習者のモデル操作に対応したソースコードを自動生成し，表示することができる．学習者が [Extra][Source Code] メニューを選択すると図 5 右のソースコードウィンドウ (SrcWin) が開く．学習者が String クラスの変数を生成すると，SrcWin の 1 行目に `String str1;` が表示される．また String クラスのオブジェクトを生成すると，`new String("Moji");` が 2 行目に表示される．ここで学習者が変数 str1 からオブジェクトにリンクを張ると，2 行目の表示が `str1 = new String("Moji");` という表示に変化する☆☆．

もし学習者がリンクを張らずに放置すると，2 行目の `new String("Moji");` はオブジェクト消滅時と同時に SrcWin から削除される．また一旦リンクを張ったあとでオブジェクトからリンクを外すと `str1 = null;` といったコードが表れる．この機能により，学習者はモデルの操作と対応するソースコード表記を同時に観察することができる．また，`null` や `new` キーワードの意味を無理なく学習することができる．通常は変数を準備してからオブジェクトを生成したほうが，オブジェクトへのリンクを張る操作に時間的余裕があるため自然であるが，先にオブジェクト生成後に変数を作成しリンクすることもできる．その場合，2 行目の `String str1;` と 1 行目の `new String("Moji");` がマージされ，`String str1 = new String("Moji");` となる．

継承関係にあるクラスを用いた場合は，参照時の「キャスト」についても必要な時のみ自動的にコードに表れる．たとえば Object の変数 obj1 から Oval のインスタンス（Oval の変数 oval1 から参照されている）にリンクする場合，生成されるコードには `obj1 = (Object) oval1;` のように，自動的にキャストが挿入される．これにより，キャストについても試行しながら学ぶことができる．またキャストがインスタンスの型を変更するのではなく，単に参照を受け渡すための記述であることも学習できる．

☆ 現在の実装では，変数を経由して，参照しているインスタンスのメソッドを呼び出す機能を設けていない．そのため，変数の型によって呼び出せるメソッドが規定されることを学習することはできない．

☆☆ 一般的には `str1 = new String("Moji");` を `str1 = "Moji";` と記述するが，ここでは新しいオブジェクト生成と new を対応させるため，あえて new を用いた記法としている．

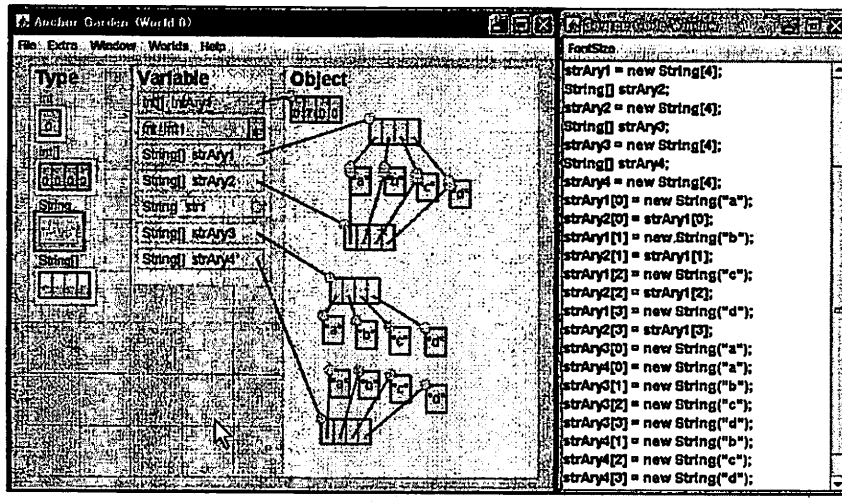

図 6　シャローコピーとディープコピー
Fig. 6　Shallow copy and deep copy.

3. 学習可能な内容

AG を用いると，以下に述べる内容を学習することができる．以下の内容は「参照」概念の理解が必要不可欠であるが，書籍や言葉による説明，サンプルプログラムの実行のみでは理解しにくい内容であるため，システムを用いることを考えた．

3.1 配列の概念

「配列」をはじめて学習する場合は，int の配列などプリミティブなデータ型の配列から始めるのが自然である．int の配列型変数を `int[] intAry1;` と定義するまでは既習概念で対応できるが，`intAry1 = new int[4];` のように配列を作成する場面で `new` キーワードが出現し，既習概念のみでは対応できなくなる．そのため学習者は `new` キーワードの意味を理解するか，“おまじない” として覚える必要がある．AG を用いると，`new` の意味や，変数と配列を=で結ぶことの意味について，文字列を作成する場合の `String str1 = new String("Moji");` との構造上の類似を確認しながら修得できる．

3.2 プリミティブの配列とオブジェクトの配列の違い

int の配列を理解した学習者が，String の配列などオブジェクトの配列を学ぶ際，はじめは int の配列と同様に “箱の中にオブジェクトが入っているイメージ” を持つものと思われる．しかし実際の配列要素は “オブジェクトへの参照” である．これを理解するためには「変数」から「オブジェクト」への “参照” 概念を理解したうえで，「配列要素は変数と同じく，参照である」ことを学ぶほうが効果的である．AG では，String の配列（インスタンス）を生成すると，リンクのタブ（円形のつまみ）が配列要素として表示される．それらのタブは String オブジェクトのみにリンクできる．こうした表示と操作上の制約により，オブジェクトの配列の構造を容易に理解することが可能となる．

3.3 シャローコピーとディープコピーの違い

前の 3.2 と関連して，配列に関して，シャローコピーとディープコピーの違いを意識してプログラミングしていないと，配列をコピーしたうえで，片方の配列要素が指すオブジェクトに変更を加えるといった場面で意図しない現象に遭遇することになる．int の配列をコピーするときは

```
for (int n = 0; n < intAry1.length; n++){
  intAry2[n] = intAry1[n];
}
```

といったコードで要素がコピーできるため問題はないが，String の配列の場合，同様のコード

```
for (int n = 0; n < strAry1.length; n++){
  strAry2[n] = strAry1[n];
}
```

では，図 6 の strAry1 と strAry2 が示す状態（シャローコピー）である．完全なコピー（ディープコピー）とするためには，

```
for (int n = 0; n < strAry3.length; n++){
  strAry4[n] = new String(strAry3[n]);
}
```

のように，新しいインスタンスを生成しなければならないことを理解する必要がある．不変オブジェクトの配列の場合は問題が発生しにくいが，内部の状態が変化する可能性のあるクラス（StringBuffer や Point など）のオブジェクトを配列で扱う段階になると，多くの場合つまずくことになる．こうしたつまずきを防ぐうえでも，プログラムにおける = の意味がプリミティブの場合とオブジェクトの場合で完全に異なることを早い段階で知っておくことが望ましい．こうした点の理解においても，オブジェクトへの参照操作から自動的にソースコードを生成できる AG が有効に作用すると考えられる．

3.4 オブジェクトのスワップ

2 つの変数が参照しているオブジェクトを交換する際の常套手段として，一時変数 temp を用意しておき，

```
temp = str1;
str1 = str2;
str2 = temp;
```

のようにする方法がある．AG を用いると，このコードが意味する事象を実際のリンク操作によって直感的に理解させることが可能となる．AG で 2 つの String 変数が，2 つの String のインスタンスを参照している状態で，それをリンクを切らないように（消滅しないように）交換するリンク切替え操作を行った例と，その操作によって生成されたコードを図 7 に示す．

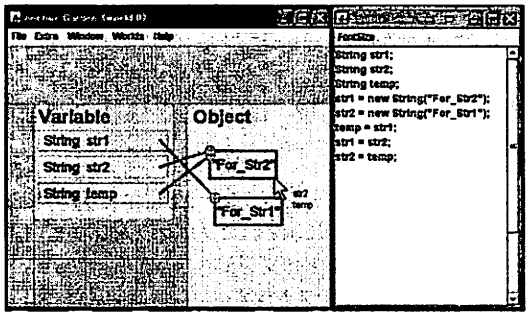

図 7 オブジェクトのスワップ
Fig. 7 Swapping objects.

4. 関連研究

これまでプログラミング初学者を対象としたプログラミング環境や支援ツールが多く開発されている．オブジェクトの振舞いを視覚化し，プログラムの理解を助ける環境として，ドリトル[5] や Alice[2),6)]，The Java Power Tools[7)]，PigWorld[8)]，Scratch[9)]，ロボットを利用した環境[10)] などがある．また Nigari[11)] は簡素だがドリトルよりも Java に近い構文を利用し，実用言語導入における段階的な概念理解を目的としている．JavaMM[12)] も Nigari に類似した簡潔記法を導入している．

ソートや 2 分木探索などのアルゴリズムを動きで示す方法として，アルゴリズムアニメーションが有効である．JAWAA[4)] は Web ベースのアニメーションを簡単に作成できるスクリプト言語である．Java アプレットで書かれた実行系が，スクリプトファイルを読み込んで表示する．Jeliot3[1),13)] は，Java プログラムの実行における変数や配列への代入，条件文などをアニメーションで表示するシステムである．配列も扱うことができ，ステップ/連続実行によりソースコードの意味を理解するうえでは有効であるが，たとえばシャローコピーなどのデータ構造を正確に図表現することはできない．

双方向リンクや 2 分木などのデータ構造を視覚化する研究についても Myers による INCENSE[14)] をはじめ，古くから行われている．jGRASP のオブジェクトビューア[15)] では，プログラム実行中の動的なデータ構造をクラス名やフィールド名から推測し，半自動的に表示できる．jGRASP が対象としている利用者は比較的高度な内容を学ぶ学習者であり，我々が対象としている初学者とは異なる．またデバッグの途中でデータ構造を観賞しながらコードを理解することに主眼を置いているため，AG のようにオブジェクトを操作するといったことはできない．

データ構造視覚化システムのなかで，我々の研究に一番関連があるのが Campbell らの LIVE（Languate-Independent Visualization Environment）[16)] である．LIVE では視覚化されたオブジェクトをインタラクティブに操作してデータ構造を学習したり，対応するソースコードを生成したりすることができる．グラフィカルな表現を用いているが，動的なオブジェクトを生成したり，リンクを張るときには変数の上でポップアップメニューから実行する形式であるため，AG のような直感的な操作を提供していない．

5. 実　　験

本システムが初学者の理解にどのような影響を与えるかを調査するため，実験を行った．

5.1 概　　要

システムを用いて学習すると，オブジェクト参照の概念への理解度や，配列を扱いながらデータを処理するプログラムを記述する能力が向上することが予想される．そのため，理解度や記述能力を測るテストの結

果についてシステム使用群と未使用群を比較することにした.

被験者は,著者が所属する大学院の修士学生(主に一年次)を対象として開講している科目「基礎プログラミング(2008年4–5月開講,90分授業,全15回)」の受講者22名とした.この科目の目的は,主に文系学部出身者にプログラミングの基礎を理解してもらうことである.講義ではC#を用いて,コンソールプログラムをVisual Studioを用いて作成する演習を含めながら進めた.

講義の構成について述べる.講義の前半では,変数やデータ型(int, String),演算子,if文,ループ(while, for)などの基本的な事項を導入した.ただし変数については,「箱のなかに値を入れる」という簡単な説明に留め,代入概念のみで説明した.講義の後半に配列や関数といった事項を導入した.配列を導入していくうえで「参照概念」が必要になるときに,初めてシステムを導入した.

5.2 手　順

講義の最初に,パソコンの使用歴やプログラミング経験などについてのアンケート調査をおこなった.人数とプログラミング経験の有無が均一化するよう,被験者を2つのグループ(A,B)に分けた.講義は表1に示す日程で行った.5月12日の1限では配列の導入を60分で行った後,B群を退席させ,A群に小テスト(1)を実施した.同日の3限ではB群のみ,システム(AG)を30分使用(起動,拡大縮小などの基本操作,配列の作成)したのち,A群と同じ小テスト(1)を行った.小テスト(1)の結果は成績には影響しないが,参考にする旨を伝えて実施した.小テスト(1)は,int配列の作成と値を格納したのち,別のint配列に逆順で格納するプログラムを作成する課題であった.被験者は問題用紙に回答した.

5月19日は1限にA群,3限にB群に対して,参照概念を導入する講義を行った.講義では値型と参照型の違いや,シャローコピーとディープコピーの違いを解説した.A群とB群で使用した講義資料(PowerPoint)は同じであったが,概念説明図についてはA群では教師が板書で説明し,学生はそれをプリントやノートに写した.B群では教師がシステム(AG)を利用してプロジェクタに映示して説明した.またB群の学生にはAGを自分で操作して体験的に学ぶよう働きかけた.60分の解説後,両グループ共通の小テスト(2)を実施した.小テスト(2)は問1〜3で構成されており,(問1)「値型」「参照型」「ディープコピー」「シャローコピー」の理解を問う○×問題,(問2)配列の作成,参照,コピーを行うプログラム中の穴埋め記述問題,(問3)問題に示されたプログラム断片を実行し終えたときの変数とオブジェクトの関係を図で示す問題であった.小テストはすべて制限時間30分として実施した.途中都合によりグループを変更した学生や,どちらかの講義を欠席した学生を除いた結果,A群は10名,B群は8名のデータを得た.なおA群が最終試験に不利になることを避けるため,5月26日に実験内容とシステム(AG)について説明した.

表1 実験授業日程
Table 1 Schedule of the Experimental Lecture

日・時限	A:未使用群	B:使用群
4/9 [3]	概説,環境設定	
4/16 [3]	計算機の構成と入出力	
4/21 [1]	はじめてのプログラミング	
4/23 [3]	変数・データ型・演算子	
4/28 [1]	条件分岐 (if)	
4/30 [3]	ループ (while,for)	
5/7 [3]	中間テスト・演習	
5/12 [1]	配列 (1):導入(代入概念のみ) 60分	
5/12 [3]	小テスト (1) 30分	—
	—	AG使用 (1) 30分 小テスト (1) 30分
5/14 [3]	配列 (2):並びかえ(代入概念のみ)	
5/19 [1]	参照概念をPPTと白板で説明 60分 小テスト (2) 30分	—
5/19 [3]	—	参照概念をPPTとAGで説明&使用 60分 小テスト (2) 30分
5/21 [3]	関数 (1)	
5/26 [1]	関数 (2) (小テスト返却)	
5/28 [3]	模擬試験 (回収・採点なし)	
6/2 [1]	最終試験	

表2 小テスト結果
Table 2 Result of Tests

小テスト (1) (満点6)			小テスト (2) (満点15)		
点数	A (人)	B (人)	点数	A (人)	B (人)
6	1	1	14-15	0	1
5	0	1	12-13	1	2
4	4	2	10-11	3	0
3	4	0	8-9	2	2
2	1	1	6-7	4	2
1	0	3	4-5	0	1
平均	3.6	3.0	平均	8.45	9.19
分散	1.16	4.0	分散	5.14	15.00

5.3 結果と考察

表2に,小テスト(1)(2)の得点分布を示す.分散が大きいが参考として平均値も掲げた.小テスト(1)(表2左)では,ちょうど上位得点と下位得点の人数が半々となった.5月12日の小テスト(1)では,AGを使用したB群のほうが若干点数が低い結果となっ

表 3 最終試験 結果
Table 3 Result of Final Test

最終試験 (全体) (満点 70)			最終試験 (問 5) (満点 20)		
点数	A (人)	B (人)	点数	A (人)	B (人)
60-70	0	4	19-20	0	2
50-59	2	0	16-18	0	1
40-49	4	1	13-15	1	2
30-39	2	3	10-12	4	1
20-29	2	0	7-9	1	0
10-19	0	0	4-6	3	2
0-9	0	0	0-3	1	0
平均	40.2	52.0	平均	8.0	13.4
分散	140.6	302.9	分散	21.1	38.8

た．B 群は説明直後にテストを受けられなかった点や，被験者の一部が直前に導入した AG に問題解決のヒントが隠れていると誤解していたことが原因と考えられる．実際には AG を使って小テスト (1) の解答プログラムの答えにたどり着けるわけではなく，用途を配列の作り方や添字の意味などに限って提供していた．

小テスト (2) (表 2 右) では，B 群のみ満点の被験者がいることを除き，両群の差はほとんど見られなかった．また小テスト (1)(2) について帰無仮説「両群の小テスト点数平均値に差がない」として t 検定を行ったが，有意差は見られなかった．原因としては AG を使用する時間が足りず，意味を十分に理解していなかったことや，AG の表現と小テストで紙に書く図との対応を理解していなかったことが考えられる．

表 3 に，6 月 2 日に実施した最終試験の得点分布を示す．最終試験は問 1〜5 で構成されており，問 1〜3 が Visual Studio を使って与えられた条件を満たすプログラムを作成する問題 (電子提出)，問 4 が小テスト (2) の問 1 と同様の○×問題，問 5 が小テスト (2) の問 3 と同様の，変数とオブジェクトの関係を図示する問題であった．配点は問 1,2,4 が 10 点，問 3,5 が 20 点であった．

表 3 左側が最終試験全体の得点分布，右側が問 5 のみの得点分布である．小テストと同様，t 検定を行ったところ，とくに問 5 の平均値の差に有意傾向が見られた (最終試験全体 $t(16) = -1.64$, p 両側 $= 0.127$, 問 5 のみ $t(16) = -2.04$, p 両側 $= 0.063$)．このことから，AG を使用し，ある程度習熟することによって，変数とオブジェクトの関係の理解度向上につながる可能性が示唆された．

表 4 と表 5 に，最終試験における各群の問ごとの得点相関を示す．両群とも問 4(○×問題) との相関が低く，問 1〜3 (プログラム作成問題) と問 5 (変数とオブジェクトの関係を図示する問題) との相関が高い傾向にある．特に比較的高度な能力が問われる問 3 と，

表 4 最終試験 Pearson の相関係数：A 群 (n=10)
(*は 5%水準 (両側)，**は 1%水準 (両側) で有意)
Table 4 Pearson's Correlations of Final Test Score

	問 1	問 2	問 3	問 4	問 5
問 1	1	0.70*	0.60	0.14	0.59
問 2	0.70*	1	0.69*	-0.23	0.43
問 3	0.60	0.69*	1	0.31	0.83**
問 4	0.14	-0.23	0.31	1	0.50
問 5	0.59	0.43	0.83**	0.50	1

表 5 最終試験 Pearson の相関係数：B 群 (n=8)
(*は 5%水準 (両側)，**は 1%水準 (両側) で有意)
Table 5 Pearson's Correlations of Final Test Score

	問 1	問 2	問 3	問 4	問 5
問 1	1	0.51	0.93**	0.17	0.76*
問 2	0.51	1	0.36	0.60	0.84**
問 3	0.93**	0.36	1	0.23	0.70
問 4	0.17	0.60	0.23	1	0.54
問 5	0.76*	0.84**	0.70	0.54	1

問 5 の相関が高い点が共通している (表 5 の [問 3][問 5] の相関 (0.70) の有意確率は.052)．このことから変数とオブジェクトの関係を図示する能力と，プログラム作成能力は互いに影響していることが確認された．今回の被験者の多くはプログラミング初学者であり，元々のプログラム能力のみが要因となって高い相関を示すことは考えにくい．よって変数とオブジェクトの概念理解を促す AG システムは，プログラム作成能力の向上に寄与する可能性が高いといえる．

6. まとめと今後の課題

静的な型付けを持つオブジェクト指向プログラミング言語におけるデータの「型」「変数」「オブジェクト」を視覚化し，さらに視覚化した各要素を操作することによって，概念理解を促進させるためのワークベンチを開発した．学習内容が型と変数，オブジェクトの関係や配列の構造に限られているが，操作が「生成」と「リンク」の 2 つだけであり，学習者にとって覚えて操作しやすいという利点がある．またワークベンチ上で表現された概念世界を直接操作できることにより，学習者が頭のなかに思い描いているモデルとの相違点を明確にでき，その差異を埋めていくことが容易になると考えられる．システムは学習者に概念世界のなかで許容された操作のみを実行させるため，学習者に誤解を与えにくい．こうしたソフトウェアを用いることにより，プログラミング学習者が実践的なオブジェクト指向プログラミング言語を学習するうえでの障壁を低くする可能性がある．また教員にとっても，本ワークベンチを操作しながら解説することで，これまで言葉でなかなか説明しづらかった概念世界について，具

体的な表現をもって提示できる利点がある.

実験の結果，AGに習熟すると変数とオブジェクトの関係の理解能力が向上する傾向が見られた．また変数とオブジェクトの関係理解能力とプログラム作成能力との相関が高いことが明らかとなり，AGがプログラム作成能力を向上する可能性が示された.

システム改良としては，ハッシュや2分木などのデータ型や，変数側からのメソッド呼び出し機能を追加することが考えられる．AGのデータ型は，視覚化のための親クラスを継承したJavaのクラスとして実現しているため，型の追加は比較的容易である．また操作を再生したり，一般のソースコード断片からアニメーションを生成できるようにしたい．またシステムによる効果の検証を継続するとともに，システム使用時間と理解度についての知見を得たいと考えている.

AnchorGardenシステムは，以下のURLにて公開している．`http://anchorgarden.mydns.jp/`

謝辞 本研究の一部は文部科学省科学研究費補助金（課題番号 20680036）の支援によるものです.

参 考 文 献

1) Moreno, A., Myller, N., Sutinen, E. and Ben-Ari, M.: Visualizing Programs with Jeliot3, *Proceedings of Advanced Visual Interfaces (AVI04)*, pp.373–376 (2004).
2) Powers, K., Ecott, S. and Hirshfield, L.M.: Through the Looking Glass: Teaching CS0 with Alice, *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp.213–217 (2007).
3) Kölling, M., Quig, B., Patterson, A. and Rosenberg, J.: The BlueJ system and its pedagogy, *Journal of Computer Science Education, Special Issue on Learning and Teaching Object Technology*, Vol.13, No.4, pp.249–268 (2003).
4) Akingbade, A., Finley, T., Jackson, D., Patel, P. and Rodger, S.H.: JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pp.162–166 (2003).
5) 兼宗 進，中谷多哉子，御手洗理英，福井眞吾，久野 靖：初中等教育におけるオブジェクト指向プログラミングの実践と評価，情報処理学会誌 プログラミング，Vol.44, No.13, pp.58–71 (2003).
6) Cooper, S., Dann, W. and Pausch, R.: Teaching Objects-first In Introductory Computer Science, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pp.191–195 (2003).
7) Proulx, V.K., Raab, J. and Rasala, R.: Objects from the Beginning — With GUIs, *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, pp.65–69 (2002).
8) Lister, R.: Teaching Java First: Experiments with a Pigs-Early Pedagogy, *Proceedings of the sixth conference on Australasian computing education*, pp.177–183 (2004).
9) Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M. and Rusk, N.: Programming by Choice: Urban Youth Learning Programming with Scratch, *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pp.367–371 (2008).
10) Lawhead, P.B., Duncan, M.E., Bland, C.G., Goldweber, M., Schep, M., Barnes, D. J. and Hollingsworth, R. G.: A Road Map for Teaching Introductory Programming Using LEGO©Mindstorms Robots, *ITiCSE-WGR '02: Working group reports from ITiCSE on Innovation and technology in computer science education*, pp.191–201 (2002).
11) 長 慎也，甲斐宗徳，川合 晶，日野孝昭，前島真一，筧 捷彦：プログラミング環境Nigari：初学者がJavaを習うまでの案内役，情報処理学会論文誌 プログラミング，Vol.45, No.9, pp.25–46 (2004).
12) Cecchi, L., Crescenzi, P. and Innocenti, G.: C : C++ = JavaMM: Java, *Proceedings of the 2nd international conference on Principles and practice of programming in Java (PPPJ2003)*, pp.75–78 (2003).
13) Moreno, A. and Joy, M.S.: Jeliot 3 in a Demanding Educational Setting, *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol.178, pp.51–59 (2007).
14) Myers, B. A.: INCENSE: A system for displaying data structures, *SIGGRAPH Comput. Graph.*, Vol.17, No.3, pp.115–125 (1983).
15) James H. Cross, I., Hendrix, T. D., Jain, J. and Barowski, L. A.: Dynamic Object Viewers for Data Structures, *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp.4–8 (2007).
16) Campbell, A. E.R., Catto, G.L. and Hansen, E.E.: Language-Independent Interactive Data Visualization, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pp.215–219 (2003).