オブジェクト指向プログラミング学習環境としての Morphic とその授業実践

伊　藤　栄　一　郎

はじめに

本稿では、オブジェクト指向プログラミング（以下、OOP と略す）の学習に、Morphic と呼ばれる GUI フレームワークを用いた演習方法について述べる。

現在、専門ゼミナール（以下、ゼミと略す）では、プログラミング言語やその方法論をテーマとし、３年次で基礎的なプログラミング言語や方法論について学び、４年次でその実践的な応用としてソフトウェア開発を行っている。スマートフォンやタブレットなどの普及により、モバイル端末のアプリケーション開発を希望する学生が増えてきた。スマートフォンのシェアの殆どを占める iPhone や Android では、Objective C や Java といったオブジェクト指向プログラミング言語でプログラムを作成するため、３年次にオブジェクト指向プログラミングパラダイムを取り上げることが以前よりも多くなった。

OOP の諸概念を理解することは重要であるが、それらの学習や教育は難しいと考えられている。[1) 2) 3)] その理由として、「オブジェクトの概念の具体的イメージがわかない」、「OOP 固有の概念を手続き型言語の延長線上で理解しようとして混乱する」[3)] といったことが挙げられている。これらの問題点に対応するために、さまざまな試みが行われている[1) 2) 3)] が、筆者はオブジェクトを視覚的に表した環境を用いて OOP の諸概念を直感的に把握すれば、OOP を効果的に学ぶ助けになるのではないかと考えている。そこで、Morphic という GUI フレームワークを利用することにより、オブジェクトやオブジェクト間の連携を視覚化し、OOP の学習に役立てたいと考えた。

オブジェクト指向プログラミング

オブジェクト指向プログラミングは、オブジェクトを中心に据えるプログラミングパラダイムの１つである。2012 年９月時点でのプログラミング言語の人気ランキング（表１）によると、１位から 10 位のうち過半の言語が、オブジェクト指向プログラミングパラダイムを採用した言語となっており、現在のソフトウェア開発において主流のプログラミングパラダイムであるといえる。

表１）プログラミング言語ランキング（2012/9, RedMonk より[4)]）

1．JavaScript（＊）
2．Java（＊）
3．PHP
4．Python（＊）
5．Ruby（＊）
6．C#（＊）
7．C++（＊）
8．C
9．Objective-C（＊）
10．Shell

（OOP をパラダイムとする言語には＊を付けている）

OOP をパラダイムとするプログラミング言語は表１以外にも多数あるが、その特徴的な概念として以下のようなものが挙げられる。

1．データ（プロパティ）と手続き（メソッド）をカプセル化するオブジェクト
2．オブジェクト間のメッセージ送受信（メッセージパッシング）
3．クラスやプロトタイプによるオブジェクト生成
4．継承や委譲
5．ポリモーフィズム

これらは OOP を標榜するプログラミング言語によって、実現方法や重点の置き方が異なっているものの、OOP 言語を習得する上で理解することが必要となる概念である。

Morphic

Morphic は Sun Microsystems においてプログラミング言語 Self のために開発された GUI フレームワークである。[5] 開発者の一人である John Maloney は、Smalltalk/80 の実装の一つである Squeak に Morphic を移植した。[6] Smalltalk/80 の GUI フレームワークとしては MVC が有名であるが、Morphic は MVC を置き換えるべく積極的に開発が続けられ、Squeak から派生した Pharo[9] や Cuis[10] などさまざまな処理系で用いられている。（図１）

MVC では、Model-View-Controller という３種類のオブジェクトが互いに役割分担して GUI を構築する。Model がドメインを、View が表示を、Controller がユーザからの操作を司り、互いにメッセージを介して協調しながらユーザとの対話を行う。ユーザからの操作は、Controller を通じて Model に伝えられ、Model に加えられた変更は、update メカニズムによって View に伝えられ、View が Model へ状態を問い合わせることで表示を変える。[6]

MVC に対して Morphic では、モーフ

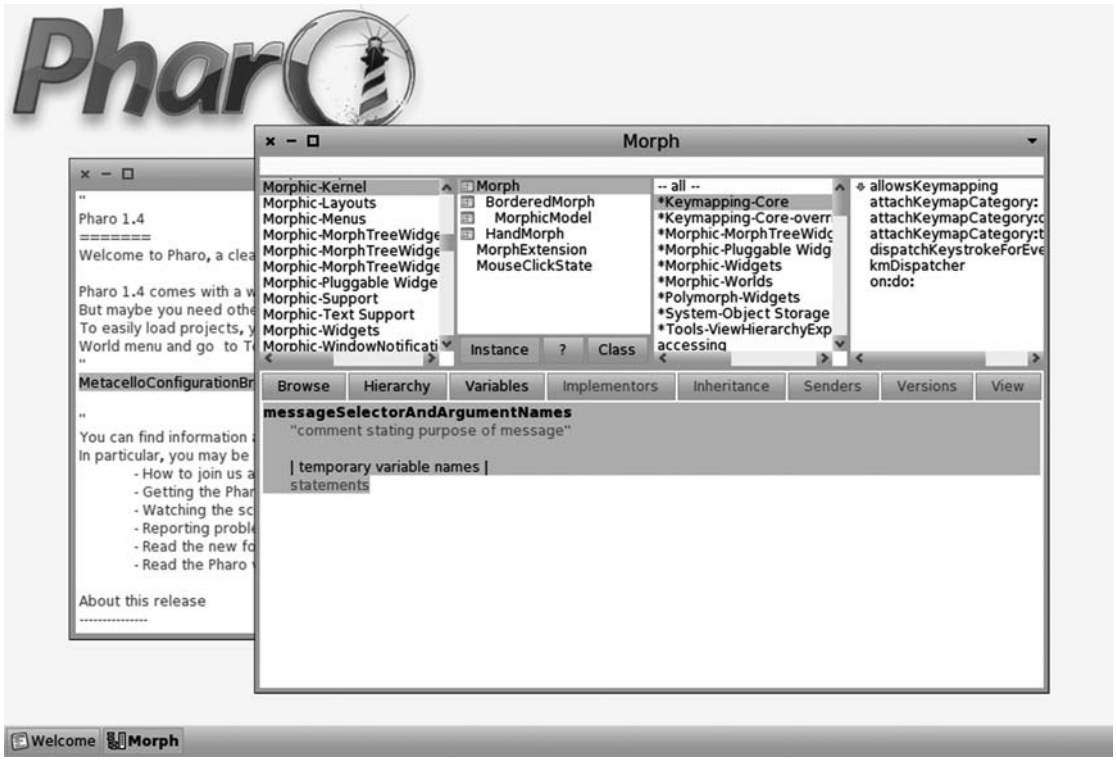

図１：Morphic フレームワークを利用した Pharo の対話的環境

（Morph）と呼ばれる特殊なオブジェクトが、MVCのViewとControllerの役割を担うことで、ユーザとの対話を一手に引き受ける。モーフはオブジェクトの内部的な状態の変化や外部からの操作に反応する視覚的な表現であり、直接操作できるだけでなく、操作に対して直ちに反応できる。[5] また、モーフは他のモーフに埋め込んで機能させることもできる。WindowsやMacなどのGUIウィジェットのように、モーフ単独では限られた能力しか持たないが、複数のモーフを埋め込み、協調して動作させることで複雑な振る舞いをさせることができる。

Morphicを使うことでユーザとの対話を行うアプリケーションが容易に構築できる。実行時にオブジェクトの内部を直接調べることも可能で、動作に不具合があった場合には、内部状態にアクセスして問題部分を修正し、動作を継続させることができる。

MorphicによるOOP演習

モーフは視覚化されたオブジェクトであり、オブジェクトの内部状態を直接的に把握できるという利点を生かせば、オブジェクト指向プログラミングの学習の初期段階において抽象的に扱いがちだったオブジェクトという概念を、モーフという具体的な形で捉えることができるだろう。単にウィジェットとして表示や操作するだけでなく、内部状態を調べたり、変えたりできるという機能を活用すれば、オブジェクトの本質を理解する強力なツールになる。そこで、2011年度からゼミにおけるOOPの学習を、Morphicを用いて行うことにした。演習を行うにあたって、以下のような学習到達目標を立てた。

1．オブジェクトを具体的にイメージできること。
2．クラスとオブジェクトの関係を理解できること。
3．プロパティやメソッドの意味や使い方がわかること。
4．メッセージパッシングによるオブジェクト間通信が理解できること。
5．オブジェクト同士の連携によってプログラムが動作することが理解できること。

これらの目標に沿って計画を立て、実施した。

ゲームを題材とした演習

2011年度のゼミではゲームを題材として演習を行った。ゲームを選んだ理由は以下の通りである。

1．学生の関心を引きやすい。
2．対象領域を理解するための時間が少なくてすむ。
3．プログラムの目的や意味を理解するのが容易である。
4．振る舞いの正誤を判定しやすい。

本格的なゲーム開発を目的にすると、高度な数学の理解や応用、チューニング技術等が必要となるため、OOPの理解という目的に沿ったシンプルなゲームを題材として選んだ。

１年間の授業期間のうち、前期に２つ、後期に１つ、合計３つのゲームを題材とした。前期は、予め準備しておいた題材を用いて、OOPの概念や技法の理解と習得を目標とした。後期は学生自身の手によってオリジナルなプログラムを作りあげることを目標とした。

JackPot

最初に題材としたゲームは、オンラインのゲーム制作雑誌「がまぐ！」（2011/２号）の記事「ゲームデザインということ」で取り上げられたJackPot（図２）である。[7] JackPotは単純な要素（プレイヤー、敵、宝物）からなる

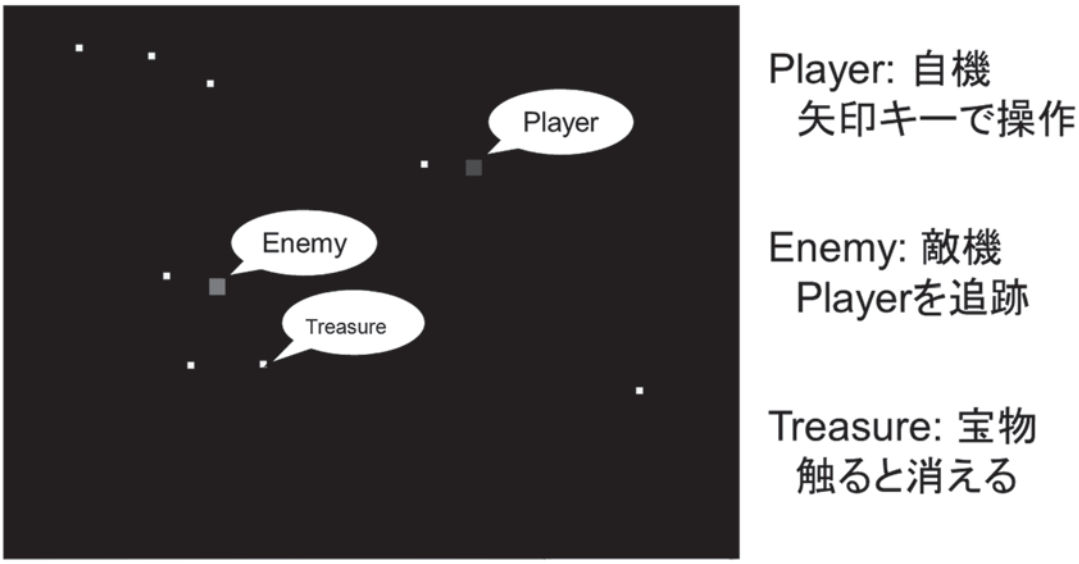

図２：JackPot ゲーム

Java 言語で記述されたゲームで、敵につかまらずに宝物を集めるというシンプルなものである。

演習の内容

元の記事は、初期版のゲームに改良を加えることでゲームを面白くしていき、ゲームデザインについて考えるという内容である。演習では記事と同じように改良をしていくが、ゲームデザインについて学ぶというより、ゲームの改良を通じてプログラムと実際の動作とを結び付け、OOP の諸概念を学ぶようにした。

具体的な内容としては、予め Morphic に移植しておいた JackPot の初期版を配布し、記述されたプログラムと実際の動作を見比べる。初期版のプログラムでは、プレイヤーと敵が同じ速度のため全ての宝物を取り終える前に最短距離で移動する敵につかまってしまう。そこで、プレイヤーの速度を２倍にする改良を行わせた。速度を２倍にするとゲームが簡単になりすぎるため、宝物を取った際の待ち時間を設けてゲームの難易度を増し、最終的には待ち時間を少しずつ増やすことで、ゲームの面白さを加えていった。

JackPot の構成

JackPot のプログラムは、クラス数が４、メソッド数が30の小さなものである（表２）。個々のメソッドも小さく、平均行数は４行以下である。画面に表示されるもの（プレイヤー、敵、宝物、ゲームの背景）は、全てモーフとしてクラス定義している。

表２　JackPot のクラス構成

実装クラス名	ゲームにおける役割	実装メソッド数
JackPot	JackPot ゲーム本体	15
JPEnemy	敵キャラクター	5
JPPlayer	プレイヤー	5
JPTreasure	宝物	5

JackPot の動作は、ゲームの本体を表す JackPot クラスの初期化（initialization）から始まる。JackPot クラスの初期化において、プレイヤー（JPPlayer）や敵（JPEnemy）、宝物（JPTreasure）のモーフを生成し、画面上に配

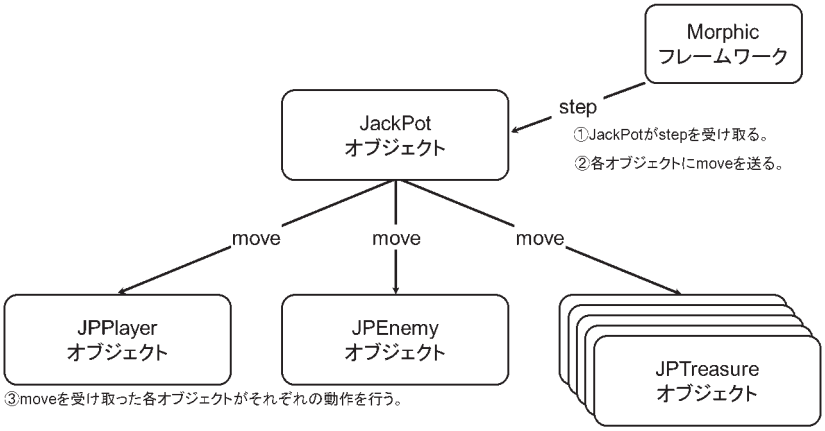

図３：JackPot の動作原理

置する。初期化後は、Morphic のフレームワークから JackPot へ定期的（１/10 秒毎）に step メッセージが送られ（図３の①）、step を受け取った JackPot は他のオブジェクトに move メッセージを送る（図３の②）。個々のモーフは、自分自身の動作に必要な処理を行うことでゲームが進んでいく（図３の③）。

JackPot のプログラム規模は非常に小さいため、プログラム全体を把握したり、個々のメソッドの役割や意味を理解したりすることが容易である。また、モーフの動きを制御する step メッセージの送信を手動で行い、少しずつゲームを進行させることができるため、どのメッセージがどんな動作を引き起こすのかを実験したり、モーフの内部状態を確認してプログラムの理解を深めたりすることができる。

JackPot で扱った主な概念は以下のようなものである。

> オブジェクトとメッセージ送信、メッセージとメソッド、クラスによるインスタンスの生成、座標、イベントと stepping、色、変数と束縛、プロパティとインスタンス変数、四則演算、ポリモーフィズム

この演習では、OOP の基本的な概念の紹介にとどめ、ゲームとプログラムとを対応づけることを主眼においた。

Danmaku

次の題材は、JackPot と同様に「がまぐ！」の記事「脳内麻薬とゲーム」で取り上げられた弾幕シューティングゲーム（以下、Danmaku と略す）である。（図４）[8] このゲームは、押し寄せてくる弾丸をプレイヤーが避け続けるという単純なものだが、反射神経が要求されるゲームである。

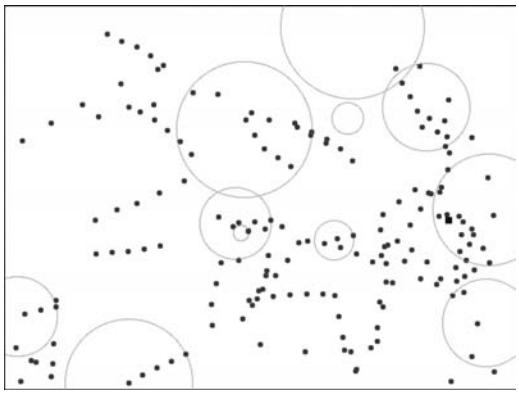

図４：Danmaku ゲーム

演習の内容

JackPot では動作するプログラムを学生に提示したが、Danmaku ではプログラムを見せず、ゲームの動作から推測してプログラムを作らせるようにした。JackPot で学んだ概念を用いてプログラムを作ることを通じ、OOP を実践的に理解させることを目指した。

作成にあたり、JackPot のプログラムの大部分を Danmaku のプログラムに流用させた。プレイヤーを追いかけてくる敵の代わりに、弾丸が一方向に進むようにプログラムを修正させた。画面上のランダムな位置からランダムなタイミングで弾丸が発射されるようにし、時間の経過とともに徐々に個数が増えるように改良して、最終的に丸い波紋から弾丸が連続して発射されるようにしてゲームを完成させた。

Danmaku の構成

前述の通り、Danmaku では学生自身がプログラム全体を作成したため、個々の学生によってプログラムに差異がある。そこで、一例として筆者が準備したデモプログラムの構成を表３に示す。JackPot と同様に４クラスで構成され、メソッドの平均行数も５行弱となっている。

表３　Danmaku のクラス構成

実装クラス名	ゲームにおける役割	実装メソッド数
Danmaku	Danmaku ゲーム本体	15
DBullet	弾丸	5
DGenerator	弾丸を生成する波紋	5
DPlayer	プレイヤー	4

Danmaku で新たに扱った概念は以下の通りである。

角度、ランダム性、三角関数、衝突判定、コレクション

この演習では、JackPot で学んだ概念を実践的に理解させることを主眼に置いた。ゲームにおけるオブジェクトの役割を明確にし、異なる役割を持ったオブジェクトを連携させることを通じて、OOP の概念を具体的に応用できるようにした。

TankBattle

当初の授業計画では、Danmaku の後にゲーム以外の実用的なアプリケーション作成を予定していたが、学園祭での展示発表として新たなゲームプログラムを作成することになった。この演習では学生のディスカッションによって開発を進めた。仕様の決定や設計も議論によって行い、実装は学生２名でペアを組んで進めるようにした。

決定した主な仕様は以下の通りである。

・ネットワークにより複数のプレイヤーが対戦する。
・プレイヤーは矢印キーで戦車を操作する。
・スペースキーによって砲弾を発射する。
・画面に現れる他のプレイヤーの戦車を攻撃し、最終的に生き残ったものが勝者となる。

作成するゲームには TankBattle という名前を付け、学園祭の来場者が楽しめるようなゲームの開発を目指した。（図５）

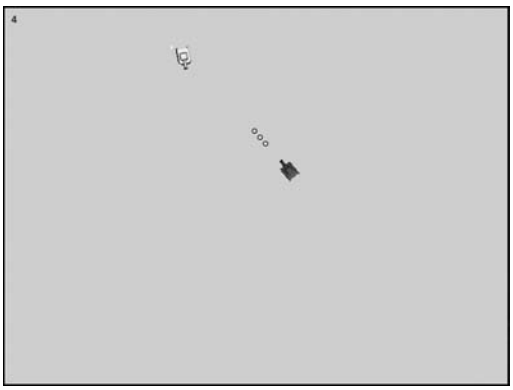

図５：学生が作成した TankBattle ゲーム

演習の内容

演習の前半は仕様を作成することに割いた。ゲームに登場する要素が増えて複雑化したため、CRC（Class Responsibility Collaborator）カードを用いてクラスの役割や関連を明確にさせた。

限られた時間の中で効率的に実装を進めるために、JackPot のプレイヤーと Danmaku の弾丸を組み合わせて TankBattle のプレイヤー側戦車を実現した。敵側の戦車については、まずランダムに動作するコンピュータ相手の対戦型ゲームを作り、最終的にプレイヤー側戦車の情報をネットワーク上に送信して、ネットワークから得た敵側戦車の情報を画面上に表示する対戦型ゲームを完成させた。

TankBattle の構成

学生がペア毎に実装を行っており、細かいクラスの内容やメソッドの規模は異なっているため、共通仕様におけるクラス構成（表４）のみを示す。

表４　TankBattle のクラス構成

クラス名	ゲームにおける役割
BattleField	TankBattle ゲーム本体
PlayerTank	プレイヤー側戦車
EnemyTank	敵側戦車
TankGun	砲弾

TankBattle で扱った新たな概念は以下の通りである。

> キーボード入力、ファイル操作、画像処理、ネットワーク、プロトコル、サブクラスとスーパークラス

この演習の時点で、数名の学生は仕様からプログラムを作成できるようになった。一度に発射できる玉に制限を加えたり、時間の経過によって玉を装填したりと、振る舞いを担うオブジェクトや必要なプロパティを特定し、求められるメソッドを追加することが可能となった。

演習の効果

Morphic を用いた演習を始めるにあたって、立てた目標は以下のようなものであった。

1．オブジェクトを具体的にイメージできること。
2．クラスとオブジェクトの関係を理解できること。
3．プロパティやメソッドの意味や使い方がわかること。
4．メッセージパッシングによるオブジェクト間通信が理解できること。
5．オブジェクト同士の連携によってプログラムが動作することが理解できること。

定量的な評価を行っているわけではなく、あくまで主観的な評価ではあるが、演習を経験した学生は、オブジェクトやクラスといった言葉を適切に用いて、提出した資料（図６）の作成やプレゼン発表を行っている。その後、卒業研究として iPhone や Android のアプリケーション開発を選んだ学生は、JackPot や Danmaku などを例に挙げながら GUI やアプリケーショ

クラス名	メソッド名	メソッドの意味や説明	呼び出すクラス
Battlefield	delet	サーバーとクライアントを止める。	Battlefield
	drawOn:	フィールドの左上に玉の数を表示する。	Battlefield
	findEnemyTank:	ネットワーク上から受け取ったIDを持つ敵がフィールド上に存在するかどうかをチェックする。	Battlefield
	handlesKeyboard:	キーボードからメッセージを受け取る。	Battlefield
	handlesMouseDown:	マウスからメッセージを受け取る。	Battlefield
	initialize	TankBattleゲームのフィールドを初期化する。具体的には、フィールドの色、大きさを初期化する。プレイヤータンクを作り、座標を初期化する。ネットワークのクライアントとサーバーを初期化する。自分自身のIDを初期化し、プレイヤータンクのIDに送る。	Battlefield
	key	キーの値を返す。	Battlefield
	keyDown:	キーボードが押された時の情報を記録する。	Battlefield
	keyUp:	キーボードが離された時の情報を記録する。	Battlefield
	mouseDown:		Battlefield
		ネットワーク上で送信された情報を受け取り、受け取った情報が	

図６：学生による TankBattle ゲームのクラス説明（抜粋）

ンモデルについて説明することができた。演習を始める前は、プログラミングに関する知識が殆どなかったことを考えると、Morphic の演習が役立っていると考えられる。

おわりに

1980 年代における Smalltalk/80 の発表から 30 年以上経った。手続き型プログラミングを理解していれば可能だった PC や Web のアプリケーション開発から、OOP の理解を必須とするスマートフォン／タブレットのアプリケーション開発へと移り、一般的な開発技術として OOP が根付いてきた。そのような背景から、大学でのプログラミング教育においても、素養として OOP を身につけることが求められるようになった。

本稿では OOP の学習環境として GUI フレームワークの１つである Morphic の利用を提案し、ゼミでの実践について紹介した。OOP の本質的な概念の理解のために、モーフという視覚的なオブジェクトを用いて、対象物の役割や意味を把握しやすいゲームを題材にするという本提案は、OOP の直観的な理解という面では効果を上げたと考えられる。今後は、学習効果について、より定量的な評価を行うとともに、非視覚的なオブジェクトの理解に結び付けることにより、効果的な OOP の教育方法の確立を目指したい。

〈参考文献〉

１）城之内　忠正,「ゲーム作りで学ぶオブジェクト指向開発」, 四日市大学環境情報論集, 2006

２）青木　浩幸他,「オブジェクト指向プログラミングの一般情報教育における意義」, 情報処理学会　研究報告, 2008

３）稲垣　宏他,「オブジェクト指向プログラミング技法のイメージ理解を目指した学習ソフトウェアの開発」, 第７回情報科学技術フォーラム, 2008

４）RedMonk, “The RedMonk Programming Language Rankings: September 2012”, http://redmonk.com, 2012

５）Maloney, J. and Smith, R., “Directness and Liveness in the Morphic User Interface Construction Environment”, ACM Eighth Annual Symposium User Interface Software and Technology, 1995

６）Mark Guzdial 他著, 軋音組訳,「Squeak 入門」, エスアイビーアクセス, 2003

７）土本　強,「ゲームデザインということ」, がまぐ！2011/２号, 2011

８）土本　強,「脳内麻薬とゲーム」, がまぐ！2011/２号, 2011

９）Pharo,「Pharo Open Source Smalltalk」, http://pharo-project.org

10）Cuis,「Cuis Smalltalk」, http://www.jvuletich.org/Cuis/Index.html