プログラム実行時のメモリ空間可視化における インタラクションの提案

小 池 伸 弥† 郷 健 太 郎††

C/C++言語プログラミングを行う際にプログラム理解支援を行うインタフェース SuZMe を開発した．本インタフェースはプログラム実行時にデバッガとして機能し，プログラムのメモリ状態を平面空間に可視化する．アドレス空間を平面空間に対応付けることで，アドレス空間に対してパンニングやズーミングなどの自由なインタラクションが可能となり，プログラムの状態を直感的に把握することができる．本論文では，可視化した平面空間におけるインタラクションを提案する．

A Proposal of Interaction on Visualization Using Address Space Representation in Runtime Program

Nobuya Koike† and Kentaro Go††

In this paper, we propose an interface, SuZMe, which helps C/C++ programmers understand programming codes. The interface works as a debugger to visualize a memory space as two-dimensional plane. It enables the programmers to navigate the data with panning and zooming functions and help him/her to promote intuitive understanding the program code. The paper discusses user interaction to objects on the visualized two-dimensional plane.

1. はじめに

日本国内の多くの情報系大学や情報系企業において，プログラミングに C/C++言語が使用されている．**図 1** は文献 1) より日本の情報系学科の大学生の学習言語の違いを示している．図によると，大学の授業で教えているプログラミング言語は C 言語がおよそ 80 %と最も高く，次いでインタプリタ言語とオブジェクト指向言語がおよそ 40 %，最後にスクリプト系言語と Web 系言語がおよそ 10 %を占めている．**図 2** は文献 1) より日本の情報系企業のプログラマの習熟した言語の違いを示している．図によると，日本の情報系企業で最も使われている言語は Java と C 言語であり，それぞれ 178 人と 153 人であった．次いで COBOL，C++，Visual Basic といった言語が使用されており，順に 50 人，44 人，24 人という人数であった．C と C++を足し合わせた習熟人数は Java の習熟人数を上回っており，企業内においても C/C++言語が広く使用されている．

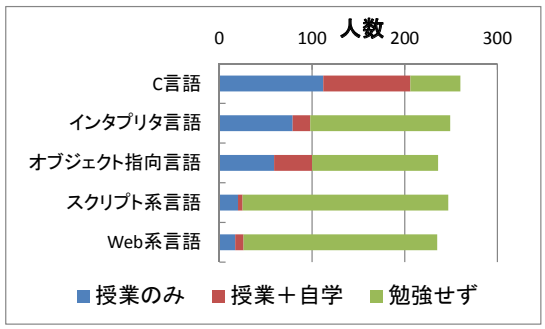

図 1 情報系学科の大学生の学習言語の違い
Fig. 1 The difference of learning languages in computer science university students

これらの結果からわかるように，日本国内の多くの情報系大学や情報系企業において C/C++言語プログラミング能力はとても重要なスキルとなっている．

1.1 C/C++言語理解の困難性

しかし，学習者が C/C++言語プログラムを理解するのには困難を要するのが実情である．**図 3** は 2010 年 10 月に山梨大学工学部コンピュータ・メディア工学科 G コース 1 年生 40 名を対象に行ったアンケートの結果である． 情報系学科に入学して半年間，構

† 山梨大学大学院 医学工学総合教育部
Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi
†† 山梨大学大学院 医学工学総合研究部
Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

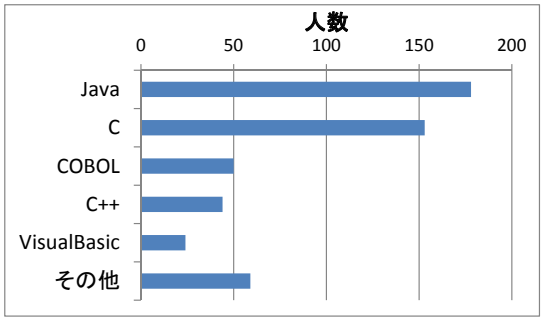

図 2　情報系企業の習熟した言語の違い
Fig. 2　The difference of proficiency languages in corporate programmer

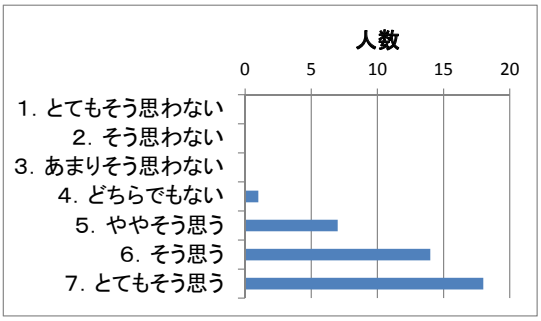

図 3　プログラミングを習うことが難しいと思うか
Fig. 3　How hard is it to learn programming?

造体やポインタなど C 言語の基礎的なことを学んだ状態で，プログラミングは難しいかという問いに対して 7 段階のリッカート尺度で回答してもらった．40 人中,「1．とてもそう思わない」,「2．そう思わない」,「3．あまりそう思わない」と答えた学生は 0 人であり,「4．どちらでもない」と答えた学生は 1 人,「5．ややそう思う」と答えた学生は 7 人,「6．そう思う」と答えた学生は 14 人,「7．とてもそう思う」と答えた学生は 18 人であった．以上の結果から，実際に C 言語を学ぶ学生が学習に苦労している様子が伺える．

C/C++言語は Java などの言語に比べると，よりハードウェアに近い低水準なプログラミング言語であるため，アドレスとポインタの概念が存在する．こういった概念をプログラミング時にうまく頭や紙に思い描くことができればよいが，それには多くの認知的意識を集中させる必要がある．また，初学者など，そもそも思い描くことができない人も存在する．これらの概念について，C 言語の教科書でもしばしば「名高い難所」[2] などと呼ばれ，C/C++言語を学習する際の 1 つの大きな山場と言われている．

1.2　解決のアプローチ

そこで本研究では，プログラムの実行過程に応じてアドレス空間を平面空間上に可視化することによってプログラムへの理解を支援する C/C++言語デバッガ SuZMe (Suitable Zooming for Memory) を開発した．本手法では目に見えないプログラムのデータ構造をグラフィカルな平面空間に表現することで，プログラムに対する認知負荷を避け，プログラムの状態を直感的に把握することができる．また，平面空間に可視化することで，パンニングやズーミングなどの自由なナビゲーションを可能とする．

2.　プログラムの一般的な特徴

2.1　プログラムのアドレス空間

プログラムのアドレス空間は非常に広大である．32bit アーキテクチャの場合，アドレス空間は 4G byte 存在し，全ての情報を画面領域内に表示することは難しい．

このように Focus+Context が重要な場合に有効なナビゲーションが Pad++[3] などに用いられている ZoomWorld[4] である．ZoomWorld は，終わりのない広大な平面上に配置された，無限の解像度を保持した情報にアクセスする，という考えに基づいている．より広い範囲を見るには，より高く，上空へと向かい，特定の項目を見るには高度を下げてコンテンツを検索し，ナビゲーションを行うというものである．そして，拡大率に応じて表示内容を変化させることを Semantic Zoom という．Google Maps☆などの既存の地図ソフトウェアの縮尺の変化には，この Semantic Zoom が使用されている．

アドレス空間に ZoomWorld や Semantic Zoom のナビゲーションを応用することで，広大なアドレス空間の中から必要な情報を取得することができる．

2.2　プログラミングの手順と 3 つのエラー

プログラマはプログラミング中に 3 つのエラーに遭遇する．プログラミングは**図 4** の手順で行われる．まず始めにソースコードをコーディングしてコンパイルを行う．その際にコンパイルエラーが発生すればその原因を見つけて修正する．コンパイルが成功すればプログラムを実行し，その際にランタイムエラーでプログラムが終了すればその原因を見つけて修正する．プログラムが正常に終了した場合はその出力結果を確認して，意図した動作と異なる結果（論理エラー）になった場合はその原因を見つけて修正する．意図した結果が出力されたらプログラマはプログラミングを終えるか，新しくコーディングを開始する．

本システムではプログラミング中に出現する 3 つの

☆ http://maps.google.co.jp/

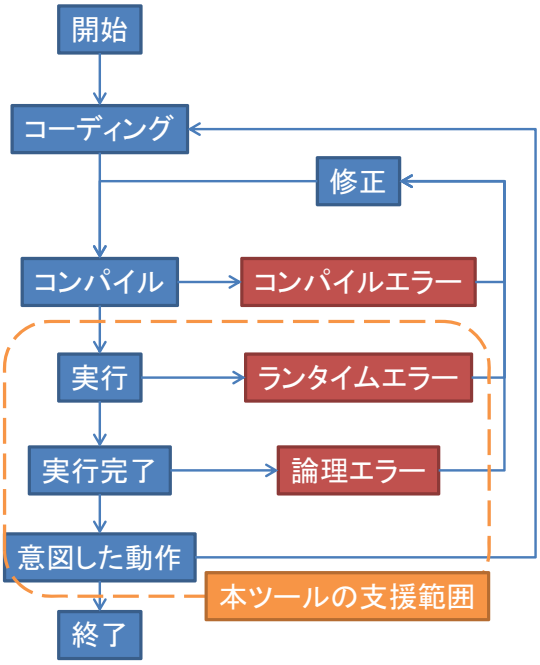

図 4 プログラミングの手順と本ツールの適応範囲
Fig. 4 The protocol of programming and range in application of SuZMe

エラーのうち，ランタイムエラーと論理エラーの 2 つのエラーに関して支援を行う．

3. 関 連 研 究

プログラミングの支援を目的とした研究は多く存在する．

プログラミング教育に関する研究として HMMMML3[5] がある．HMMMML3 にはソースコード内に多少のスペルミスがあっても，好意的に解釈してコンパイルを行うコンパイラが付属している．コンパイル後に間違っていた場所を教えてくれることでプログラミング学習に対するモチベーションの低下を防いでいる．これは，前述した 3 つのエラーの中のコンパイルエラーに対応した研究である．

プログラムの 3 つのエラーの中のランタイムエラーと論理エラーに対応するツールが，DDD☆や GVD☆☆などに代表されるようなデバッガである．デバッガとはプログラムが正しくふるまわない理由を調べるのに役立つソフトウェアツールである．デバッガはプログラム中の動的に変化していく処理や変数の値を視覚的に表示することでエラーの原因特定の手助けを行う．DDD や GVD はそれぞれ「Canvas」や「Data Window」という部分に各データノードをボックスで抽象的に表し，ポインタによって互いに関係するノードを矢印で示している．

プログラミング支援の中でも，プログラムに関する情報を視覚的に表現することをプログラム可視化（ソフトウェア可視化）という．Myers はプログラム可視化を静的コード可視化システム，静的データ可視化システム，動的コード可視化システム，動的データ可視化システムの 4 つに分類した[6]．これらは対象の可視化表現が変化を伴わないかどうか，また，可視化対象がプログラム中のコードであるかデータであるかによって分けられている．

動的な可視化表現の中でもアニメーションを使いアルゴリズムを視覚的に示すことをアルゴリズムアニメーションという．BALSA[7] は各種アルゴリズムをアルゴリズムアニメーションで可視化している．例えばバブルソートプログラムは，各データの値に対応する高さを持った柱として表示され，プログラムの実行とともにデータ同士が交換されていく様子が動的に表示される．

以下では，提案手法に関連する研究を述べる．

3.1 VINCE

VINCE[8] は C プログラムの実行をグラフィカルに表示するチュートリアルツールである．VINCE はアドレスの状態を可視化しており，1 バイトを正方形 1 つとして配置し，変数がそのアドレスに割り当てられるとその変数が必要とするサイズ分の正方形が固まりになり，変数の値が表示される．

3.2 Jeliot 3

Jeliot 3[9] は変数をグラフィカルに可視化する動的データ可視化システムのデバッガである．アニメーションを用い，ソースコードの実行に合わせて変数の変化をグラフィカルに表現することで，学習者にもわかりやすいように工夫されている．

3.3 Code Canvas

Code Canvas[10] は膨大なコードの中から空間表現を用いて関係性をわかりやすく可視化する動的コード可視化システムである．Code Canvas は ZoomWorld や Semantic Zoom のナビゲーションを使用しており，既存の地図ソフトウェアと似たようなインタラクションが行える．大量のコードが表示されているキャンバスをシームレスにズームし，空間上に配置された関連するコードを矢印で結ぶことで，コードの関係性を図で把握することができる．これに似た手法としては Code bubbles[11] が挙げられる．

☆ http://www.gnu.org/s/ddd/
☆☆ http://directory.fsf.org/project/gnuVisualDebugger/

表 1　可視化手法の特徴比較
Table 1　Comparison of visualization methods

	コード可視化	データ可視化	アドレス可視化	Zoom World の使用
VINCE	△	△	△	−
Jeliot 3	△	○	−	−
CodeCanvas	○	−	−	○
SuZMe	△	○	○	○

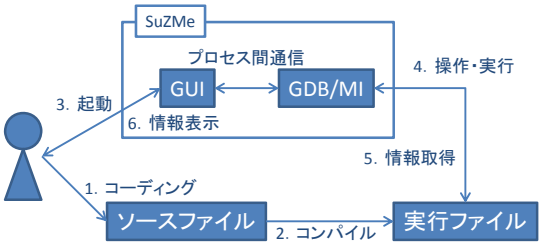

図 5　システム構成と使用手順
Fig. 5　System structure and usage

4. SuZMe

4.1 設　　計

SuZMe は作成したプログラムに対してデバッガとして機能し，アドレス空間を平面空間に可視化する．アドレスの番地と空間上の位置を対応付けて表示することで，プログラムの状態に対して直感的な把握が可能になる．

人が何かを理解する時，具体性の程度が重要になるケースが多い．例えば,「0xbffea000 から 0xbffff000 までがスタック領域」と言われるよりも「アドレス空間全体をこの線とすると，この後ろのあたりがスタック領域」と抽象化して言われた方が理解しやすい．また，逆に「ポインタをインクリメントするとデータ型のサイズ分，値が増加する」と言われるよりも「0xb7fe1000 を指している int 型ポインタをインクリメントすると 0xb7fe1004 になる」と具体的な例を用いて説明した方が理解しやすい場合もある．利用者の興味に応じて具体性の程度を柔軟に変化できるようにすることで，より広い状況に対応できる．本提案手法では，平面空間に Semantic Zoom を利用することで具体性の程度を変化させる．

4.2 従来手法との比較

従来手法との比較を**表 1** に示す．VINCE はアドレスの可視化を行っているが，単に正方形で表示されているだけであり，その正方形の位置に情報がある訳ではない．Jeliot 3 はアニメーションを加えてグラフィカルにデータを可視化しているが，アドレスの概念がない Java 言語のデバッガであるためアドレスの可視化機能はなく，変数データの空間的位置には物理的な意味はない．Code Canvas は ZoomWorld を使用したソフトウェア可視化という点で本提案手法と似ている．しかし，Code Canvas はコードを可視化する動的コード可視化システムであるのに対し，本提案手法はデータを可視化する動的データ可視化システムである点が大きく異なる．

4.3 実装と使用方法

システムの構成と使用手順を**図 5** に示す．

デバッガとしてプログラムから使用する用途に最適化された GDB☆のインタフェースである GDB/MI を使用し，プロセス間通信を経由して情報を取得する．その他のプログラムは Qt フレームワーク☆☆を使用して C++で実装した．

システムの使用手順は以下の通りである．まず始めに使用者はソースファイルを書き，それをデバッグ情報を付加してコンパイルする．次にシステムを起動し，そこからコンパイルした実行ファイルを実行する．そして，GDB/MI を経由して手に入れたプログラム情報を平面空間に可視化することで，使用者はプログラムの状態を知ることができる．

5. 可視化とインタラクション

図 6 はメモリ空間を可視化した平面空間である．アドレスを平面空間に一直線に可視化して，対応するアドレスの値を表示している．該当アドレスが変数として確保されている場合は，そのアドレスの場所に変数オブジェクトが表示され，値が更新されると枠が赤く強調される．黒線の長方形で囲まれた部分は関数のスタックを示している．ビューの最下部には Seesoft[12] を参考にしてアドレス全体を 1 本のラインとして表示し，現在のビューの位置を黒の逆三角形で示している．

アドレス空間を表示するビューは「なめらかなインタフェース」[13] としてインタラクションが可能である．以下ではこの平面空間におけるインタラクションについて述べる．

5.1 パンとズーム

この空間は ZoomWorld ナビゲーションを採用しており，マウスドラッグでパンニング，マウスホイールでズーミングが可能になっている．ズーム時は連続性を持ってなめらかに拡大縮小を行う．

5.2 Semantic Zoom

変数オブジェクトの表示は拡大率に応じて変化する．

☆ http://www.gnu.org/software/gdb/
☆☆ http://qt.nokia.com/products/

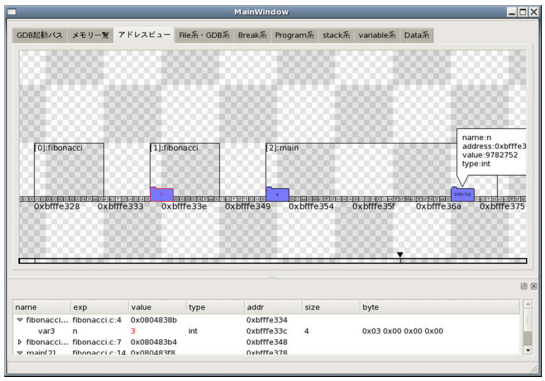

図 6　可視化された平面空間
Fig. 6　Visualized space

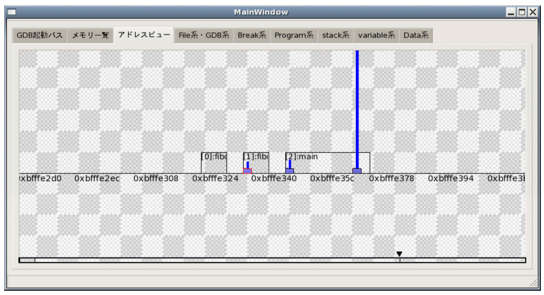

図 7　縮小した平面空間
Fig. 7　Zoomed out space

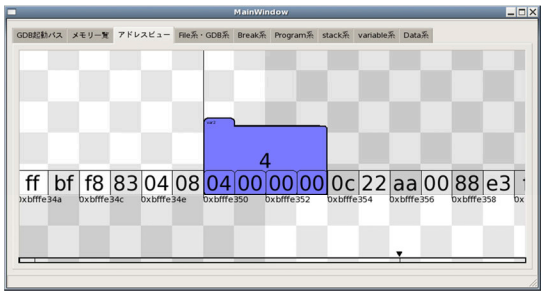

図 8　拡大した平面空間
Fig. 8　Zoomed in space

通常では変数オブジェクトにはその変数の値が表示されているが，ズームアウトすることで**図 7** のように変数オブジェクトの上下に変数の値の高さに応じたバーが表示される．逆にズームインすることで**図 8** のようにバイト列単位での表示になる．さらにズームインすることで**図 9** のようにビット列単位での表示になる．

5.3　変数オブジェクトのインタラクション

変数オブジェクトを対象としたインタラクションを**図 10** に示す．対象をクリックすると，その上部に詳細情報が表示された吹き出しの表示/非表示が切り替わる．対象をドラッグすることで空間上の任意の場所に移動できるが，対象とアドレスとはバネのようにつながっており，ドロップすると元のアドレスの位置に戻ろうとする．ドラッグ中にビューの左上に表示されるピンに対象をドラッグして触れさせることで対象にピンが刺さる．ピンを刺すことでバネの力を受けなくなり，空間上の任意の場所に対象を配置することが可能となる．ピンをクリックすることでピンが消え，再び元のアドレスの位置に戻ろうとする．

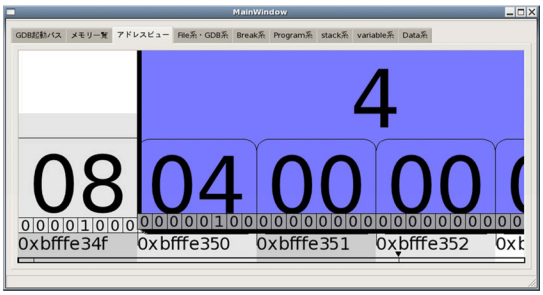

図 9　さらに拡大した平面空間
Fig. 9　More zoomed in space

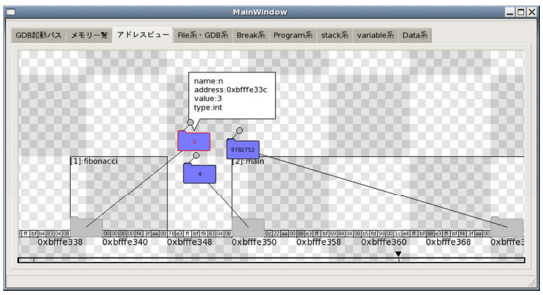

図 10　変数オブジェクトのインタラクション
Fig. 10　Interaction for variable objects

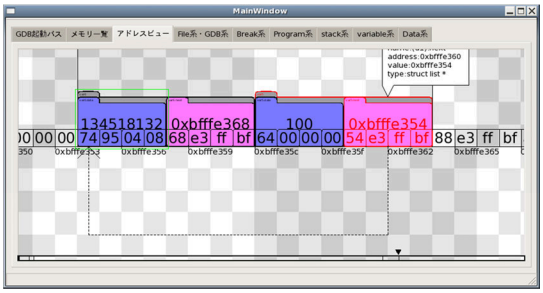

図 11　ポインタ変数オブジェクトのインタラクション
Fig. 11　Interaction for pointer variable objects

5.4　ポインタ変数のインタラクション

変数オブジェクトがポインタ変数の場合には，**図 11** のようにクリックによりポインタ先のアドレスを矢印で指し示す．また，この矢印も対象先とバネのようにつながっており，ポインタを引っ張ることで納豆ビュー[14] のように関連する変数オブジェクトがつられて引っ張られる．

6. 適　応　例

これらの可視化とインタラクションにより，1 つのビューで以下の確認が容易になる．

特定の変数を集めて変化の確認　ピンを使い，目的の変数オブジェクトを１つの場所にまとめることで，遠く離れた変数同士を見比べることができる

アルゴリズムアニメーション　配列の値をバーで見えるようにズームアウトした状態にしてソートアルゴリズムを実行することで，ソートの様子を確認することができる

エンディアンの違い　変数にズームインし，バイト列単位の表示にしてリトルエンディアンとビッグエンディアンの違いを確認することができる

ビット演算の様子　ビット列単位の表示にしてビット演算の結果を確認することができる

ポインタがどこをさしているか　ポインタがどこを指しているか図で確認できる

ヒープ領域とスタック領域の違い　malloc 関数などで取得されるアドレスがスタック領域から離れた位置にあることが図で確認できる

7. おわりに

本研究では，C/C++プログラム理解支援を行うデバッガ SuZMe の可視化したアドレス空間におけるインタラクションを提案した．SuZMe は ZoomWorld ナビゲーションを使用することで，利用者の興味に応じてプログラムの柔軟な表示が可能となる．そして，プログラムに対してより直感的なインタラクションができるようになった．以上のことから実行時のプログラムの状態を把握しやすくし，使用者の負担を軽減することができる．

今後はこの平面空間での更なるインタラクションを考えていきたい．特に，機械語に翻訳されたプログラムが配置されているテキスト領域に関して，コードの可視化ができないか検討している．また，プログラミングを行う人に広く使ってもらい，意見を元にフィードバックを行っていきたい．

参考文献

1) IT 教育実態調査プロジェクト：大学等におけるＩＴ教育実態調査報告書（情報系学科卒業生の視点），調査報告，経済産業省 (2004).
2) 小林健一郎：カーニハン＆リッチー「プログラミング言語 C」を読む，講談社 (2006).
3) Bederson, B.B. and Hollan, J.D.: Pad++: a zooming graphical interface for exploring alternate interface physics, *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, New York, NY, USA, ACM, pp.17–26 (1994).
4) ジェフ・ラスキン，村上雅章：ヒューメイン・インタフェース　―人に優しいシステムへの新たな指針―，株式会社ピアソン・エデュケーション (2001).
5) 中橋雅弘，宮下芳明：HMMMML3:他人を意識したモチベーション向上を考えたプログラミング環境，インタラクション 2011，pp.511–514 (2011).
6) Myers, B.A.: Visual programming, programming by example, and program visualization: a taxonomy, *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '86, New York, NY, USA, ACM, pp.59–66 (1986).
7) Brown, M.H. and Sedgewick, R.: A system for algorithm animation, *SIGGRAPH Comput. Graph.*, Vol.18, pp.177–186 (1984).
8) Rowe, G. and Thorburn, G.: VINCE:an online tutorial tool for teaching introductory programming, *British Journal of Educational Technology*, Vol.31, pp.359–369 (1998).
9) Moreno, A., Myller, N., Sutinen, E. and Ben-Ari, M.: Visualizing programs with Jeliot 3, *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, New York, NY, USA, ACM, pp.373–376 (2004).
10) DeLine, R. and Rowan, K.: Code canvas: zooming towards better development environments, *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, New York, NY, USA, ACM, pp.207–210 (2010).
11) Bragdon, A., Zeleznik, R., Reiss, S.P., Karumuri, S., Cheung, W., Kaplan, J., Coleman, C., Adeputra, F. and LaViola, Jr., J.J.: Code bubbles: a working set-based interface for code understanding and maintenance, *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, New York, NY, USA, ACM, pp.2503–2512 (2010).
12) Eick, S.G., Steffen, J.L. and Sumner, Jr., E.E.: Seesoft-A Tool for Visualizing Line Oriented Software Statistics, *IEEE Trans. Softw. Eng.*, Vol.18, pp.957–968 (1992).
13) 増井俊之，水口　充，Borden, G.，柏木宏一：なめらかなユーザインタフェース，第 37 回冬のプログラミング・シンポジウム予稿集，pp.12–23 (1996).
14) 塩澤秀和，西山晴彦，松下　温：「納豆ビュー」の対話的な情報視覚化における位置づけ，情報処理学会論文誌，Vol.38, No.11, pp.2331–2342 (1997).