インフォグラフィックスを用いたソースコードの可視化

Visualization of Source Code Using Infographics

岩田 まどか[*1], 隼田 尚彦[*2], 向田 茂[*2], 齋藤 一[*2], 安田 光孝[*2]
Madoka IWATA[*1], Naohiko HAYATA[*2], Shigeru MUKAIDA[*2], Hajime SAITO[*2], Mitsutaka YASUDA[*2]
[*1] 北海道情報大学大学院経営情報学研究科
[*1] Graduate School of Management and Information Sciences, Hokkaido Information University
[*2] 北海道情報大学情報メディア学部
[*2] Faculty of Information Media, Hokkaido Information University
Email: s1381109@gmail.com

あらまし:プログラミング学習の支援方法としてプログラムの可視化, ビジュアルプログラミングがある. しかし, 全くの初心者を対象としたプログラムソースの適切な可視化は少ない. また, ビジュアルプログラミングからテキストプログラミングへの移行は, 記述形式が異なるため, 関連づけが難しい場合がある. 本研究では, インフォグラフィックスを用いたソースコードの可視化を提案し, プログラムソースとの関連づけを試みる.
キーワード: アルゴリズム理解, ビジュアルプログラミング, 可視化, インフォグラフィックス

1. はじめに

1.1 研究背景

本学では, CG やインタラクティブな技術を用いるプログラミングの講義がある. それらの講義で得た知識は, 卒業研究の作品制作に活かされている. グラフィカルな要素やインタラクティブな要素を取り入れた作品は年々増加する傾向にある.

しかし, 美麗なグラフィックやインタラクションからプログラミングに興味が湧いても, ソースコードをなかなか理解できない場合がある. そのためプログラミングに対して苦手意識を持ってしまい, 作品制作に苦労する学生がいる. 文字の羅列から拒絶反応を示す学生もおり, そういった学生のプログラミングに対するハードルを下げる工夫が必要となる.

1.2 先行事例と問題点

プログラミング学習者を支援する研究として, プログラムの可視化がある. Moreno らの開発した Jeliot3 [(1)] は, 変数の遷移や関数の呼び出しなど, プログラムの流れをアニメーションで表現し, 学習者の理解を支援する. テクマトリックス株式会社の製品 Understand [(2)] は, 制御フロー, 変数・関数の呼び出しなどをダイアグラムやグラフで可視化し, 大規模なプログラムの解析を容易にする.

また, プログラムの可視化から派生したビジュアルプログラミングの研究がある. ビジュアルプログラミングは図や矢印を用いるので, テキストプログラミングの知識が乏しくてもプログラムを作れる. データフロー型のビジュアルプログラミングでは, LabVIEW [(3)] や Max [(4)], 制御フロー型のビジュアルプログラミングでは, Scratch [(5)] や Etoys/Squeak [(6)] などが挙げられる.

しかし, プログラムの可視化では, 個々の処理内容を文字で表現するものが多く, 現実とリンクさせたイメージに置き換えることを難しく感じる学習者がいる. また, ビジュアルプログラミングを理解できても, 記述形式が異なるために, テキストプログラミングへ関連づけることが難しい学習者もいる.

1.3 研究目的

本研究では, 対象学生がグラフィックに興味を持っていること, より具体的なイメージを想起させる可能性があることからインフォグラフィックスを利用したソースコードの可視化を試みる. グラフィカルでインタラクティブなプログラムに興味を持つが, プログラミングに対して苦手意識のある学生を研究対象とする. 対象学生がソースコードを理解するために, 本研究の可視化手法が寄与できるのか, プログラミングに対する苦手意識を軽減できるのかを調査する.

2. 可視化概要

2.1 可視化の対象

本研究では, グラフィカルでインタラクティブな要素を含むプログラミングを前提とし, ソースコード内で利用頻度の高い構成要素を可視化する. 変数,

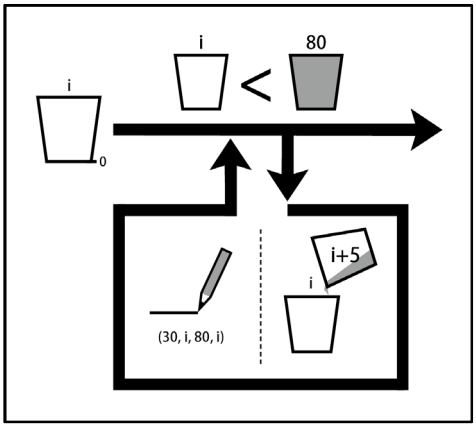

図 1 繰り返し処理を可視化した例

配列，制御構造，図形描写，画像処理，入出力処理を変換対象とする．個々の要素をピクトグラムに置き換え，プログラムの流れを 1 つのインフォグラフィックスに変換する．図 1 は，繰り返し処理をオーソドックスに可視化した例である．

2.2 可視化する言語

Java で書かれた Processing という統合開発環境を利用する．Processing は，グラフィック描写やインタラクティブな操作を Java よりも平易なコードで実装できる．様々な大学の CG を学ぶ講義でも使用されている．

3. インフォグラフィックスの実験

3.1 実験概要

制作したインフォグラフィックスから意味を読み取れるか，ソースコードとの関連性を感じられるか，検証を行う前に 1 時間程度の予備実験を実施した．本学のデザインやプログラミングを勉強している学生 8 名からアンケートを取り，インタビューを行った．被験者は Processing の利用経験があることを前提としている．今回は，プログラミングスキルが高い学生にも実験に参加してもらい，提示する可視化の善し悪しについて質問した．

3.2 実験の仮説

本研究では，インフォグラフィックスからプログラムの処理内容を読み取れる．もしくは，ソースコードのみでは意味のわからない処理でも，インフォグラフィックスと見比べることで処理内容を理解できるという仮説を設定した．

3.3 実験の手順

始めに，Processing のスキルやプログラミングの苦手分野の確認を目的とするアンケートを行う．次に，同じ意味を持つがデザインの異なるインフォグラフィックスを 4 つ並べた紙を提示し，全体を通してインフォグラフィックスの意味を考えさせる．回答方法は自由記述・口述から選択させる．回答時には，選択理由や提示したインフォグラフィックスについて尋ね，そのやり取りを録音している．回答後はインフォグラフィックスに対応するソースコードを見せ，被験者の理解と一致しているか確認を取る．もし答えられなかった場合は，ソースコードとインフォグラフィックス両方を提示し，処理内容を説明する．意味を考えるとき，意味を教えられたときのいずれも，参考になったインフォグラフィックスに順位を付け，参考にならなかったものにはバツを付けさせる．意味とデザインを変えた同形式の問題を 9 問解かせ，最後に同じ処理の流れを表す 10 種のデザインをわかりやすい順に順位付けさせる．

3.4 実験の結果と考察

変数や配列の可視化では，コップに水を入れてメモリで値を示す表現と箱に値の書かれた丸いものを入れる表現が選ばれる傾向にあった．被験者からは，実際に値を出し入れしているイメージを持ちやすいというコメントがあった．

色の指定では RGB を表す絵の具を 3 色並べることで，実際の色をイメージしやすいという意見が多数寄せられた．

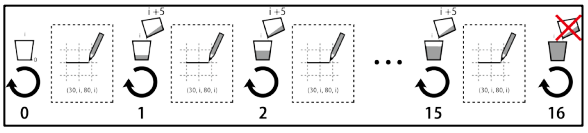

図 2 繰り返し処理を可視化した例

繰り返し処理に関しては，デザインを学びプログラミングを苦手とする被験者が，ソースコード表記に合わせた図 1 の表現よりも繰り返し処理を毎回記述する図 2 の表現を選ぶ傾向にあった．図 1 では繰り返しの表現は読み取れるが，処理内容は図 2 のような表現がわかりやすいという発言が多く見られた．このことから，ロジックを理解できない学生には，毎回どのような処理が行われているか表記された形式がわかりやすい傾向にあると考える．

実験では，文字情報のみではなく現実世界とリンクした表現を取り入れることで，プログラミングが書けない学生も処理の内容を説明することができた．その後，ソースコードを見せて説明することで，初めて概念を理解できた学生も複数見られた．

実験後，被験者達は「図であれば意味を理解できる」，「楽しく勉強できそう」，「面白い」といった感想を述べた．以上のことから，プログラミング学習時に適切な図を提示することで，被験者の理解を補助でき，苦手意識を和らげる効果を得られる可能性があると考える．

4. おわりに

本稿では，インフォグラフィックスを用いたソースコードの可視化を検討し，インフォグラフィックスから意味を読み取れるか，ソースコードと関連づけられるか予備実験を行った．今後は，被験者数を増やし，多くの実験結果から回答の傾向を見いだしたい．そして，実験結果を基にインフォグラフィックスを改良し，様々な処理パターンにおいてどの可視化が適切なのか検証を行う予定である．

参考文献

(1) Jeliot3, http://cs.joensuu.fi/jeliot/index.php (参照 2014.06.18)
(2) テクマトリックス株式会社：“Understand”, https://www.techmatrix.co.jp/quality/understand/index.html(参照 2014.06.18)
(3) 日本ナショナルインスツルメンツ：“NI LabVIEW”, http://www.ni.com/labview/ja/(参照 2014.06.18)
(4) cycling74：“Max”, http://cycling74.com/products/max/ (参照 2014.06.18)
(5) Scratch, http://scratch.mit.edu(参照 2014.06.18)
(6) Squeakland :“Etoys/Squeak”, http://squeakland.org(参照 2014.06.18)