視点移動を配慮した Jeliot3 の改良

07H092 福井俊雄

目次

概要

プログラミング導入教育でプログラミングの基礎部分でつまづく学生が多い。その原因はプログラムの動作を理解できないからであり、この問題によって学習意欲が低下する問題も併発している。プログラミング教育をサポートするツールとして Jeliot3 がある。Jeliot はプログラムの動作を可視化することでプログラムの動作をアニメーションにする事ができる。また、アニメーションの動作をユーザーの好みに合わせて変化させることができるので、学習しやすい環境になっている。しかし、現在の Jeliot のシステムではソースコードとアニメーションの二つに視線を合わせないとプログラムの内容を理解することができない事や条件分岐やループを行った時にプログラムが実行している階層をアニメーションでは読み取ることができない。そこで利用者の視点をアニメーションが行われているしあたー部分に集中させる事で負担を減らしつつ、条件分岐やループの時に階層を読み取る事ができるアニメーション改良を考案した。提案ではシアターに視点を集中させるためにアニメーションで説明している部分のコードをシアター内に提示する。また、CONSTANTS エリアを廃止してコードから変数を作成させ、視点の移動を抑えるアニメーションにした。条件分岐やループを行った時は階層が読み取れるためにウインドウを展開し、重ね合わせる事で階層表現を行った。この事によって学習者は必要な情報を得てプログラムの動作を理解しやすくなり、プログラミングの理解が深まる。今後の課題は実際にこのシステムを構築し、学習者に実験することである。

1　はじめに

プログラミング導入教育においてプログラミングの基礎部分でつまづく学生が多い [1]。その原因はプログラミング言語で書かれたプログラムが実際にどのようなアルゴリズムで動いているかを理解できないからである [2]。また、つまづく事によって学習意欲の低下や学習自体をやめてしまう問題もある。そこでプログラムがどのように動作していて、思い通りに動作をさせるためにどのような事を行えばいいのかを理解できれば問題を解決できると考えた。そのためにプログラムの動作をアニメーションによって可視化する事でわかりやすくプログラムの動作を説明するものを初学者に提示する事を考えた。プログラムの動きをアニメーションにするツールとして図 **1** の Jeliot3 というツールがある。このツールは Java のソースコードからプログラムの動作を自動でアニメーションにする。Jeliot3 では図 **2** からコントロールエリアにあるバーを操作することでアニメーションの速度を調節する事やアニメーションの履歴を確認できるので、ユーザーが学習しやすい環境になっている。しかし、このツールではプログラムの動作を必要最低限のアニメーションだけで表示しているので初学者にとっては理解しにくい。また、Jeliot3 ではソースコードとアニメーション部分が別になっているために学習者の視点が集中できず、学習しにくいと考えた。そこで初学者が学習しやすいアニメーションの改良を提案した。

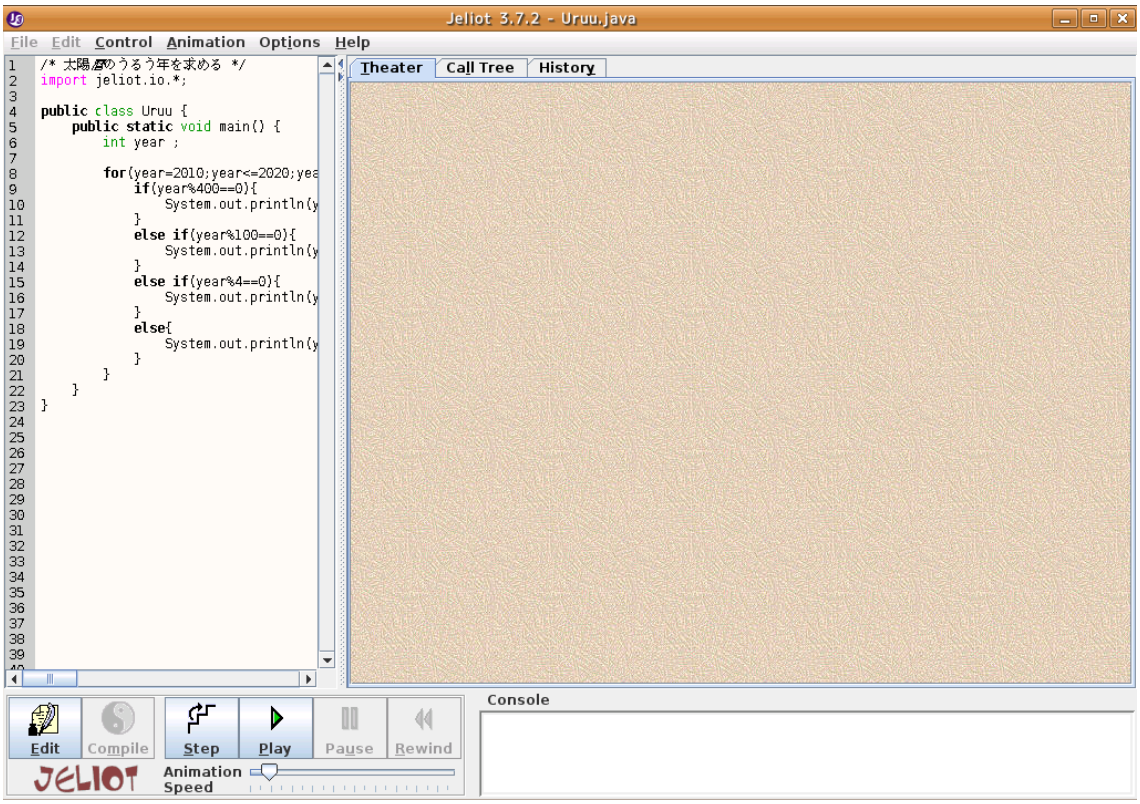

図 1　Jeliot3 のメイン画面

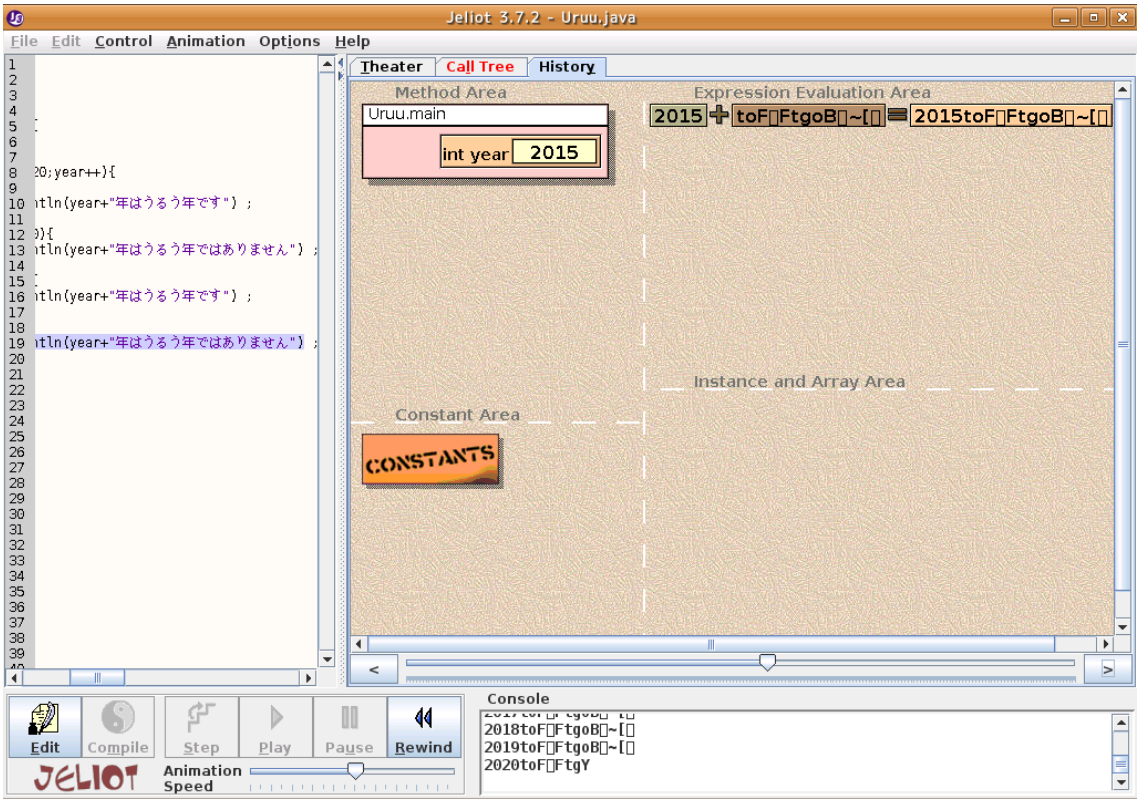

図 2　Jeliot3 はステップ再生やスピード調節、履歴を使うと学習速度を調節できる

2　目的

2.1　研究背景

プログラミング初学者は教材などに書かれているプログラムがどのように動作するのかイメージできないので、教材に書かれている文章や出力結果だけでは理解できない。また、プログラムが理解できない事によって初学者が苦手意識を持ってしまい学習のモチベーションが下がるという問題が併発している。問題の原因としてはプログラムの結果をイメージできない事や学習初期に達成の低さから起こる挫折によるものである [1,3]。これらの問題を解決するためにプログラムの動きを可視化して学習者がプログラムの動きを理解しながら学習できるようにする。プログラム動きを可視化するツールとして Jeliot3 [4,5] がある。本研究では Jeliot3 が持っている問題を考え、その問題を解決する方法を提案した。

2.2　Jeliot3 について

Jeliot3 とはプログラムの動きをアニメーションとして可視化するツールである。このシステムは図 **3** のように Java のソースコードから自動でプログラムの動作をアニメーション化する。Jeliot3 の構成はソースコードのエディター機能、アニメーションを実行するシアター部分、プログラムからの出力を表示するコンソール部分がある。エディターの部分にはソースコードを自分で記述する事ができる。また保存している Java ソースコードを展開してアニメーションにできる。アニメーションを実行している時、シアターの動作に相当するソースコードはハイライトされて現在どの部分を行っているかを把握しやすく設計されている。アニメーションのスピードを調節する事やステップ再生をする事によって学習者が学習しやすいスピードにできる。この事によって Jeliot を利用した学習者はプログラミング学習を効果的に行うことができる。

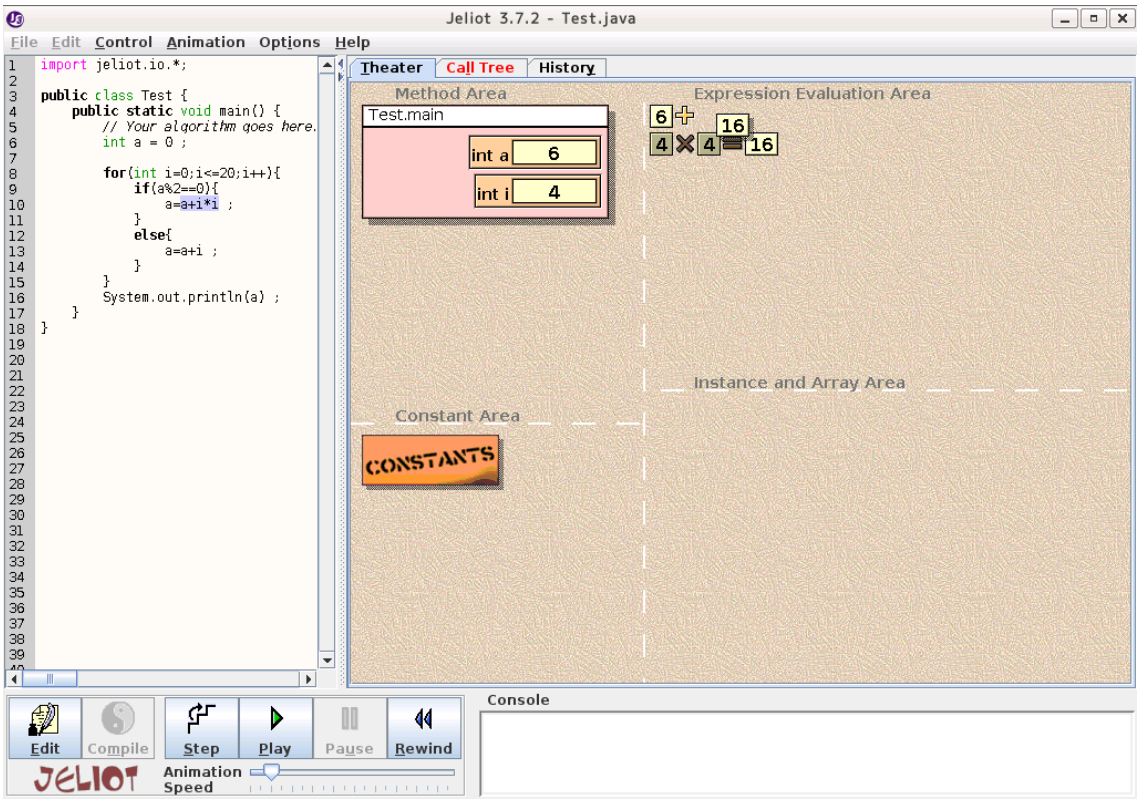

図 3　Jeliot3 での演算

Jeliot3 の始まりは University of Helsinki で開発された Eliot [4, 5] からである。Eliot は C 言語で書かれたプログラムをアニメーションにするものだった。次に Jeliot が University of Helsinki で開発された。これは Eliot が X Window System のある環境でしか動作しなかったので様々なプラットフォームで動作させるために開発された。Web プラットフォームにあわせるために開発言語が Java に代わった。Java で作られたため Java で書かれたブログラムをアニメーションにするシステムに変わった。その後、ユーザーインターフェース改良した Jeliot2000 が Weizmann Institute of Science で開発された。Jeliot3 では Jeliot2000 でアニメーションエンジンと Java インタプリタが強い依存関係を持っていた問題を二つを別のシステムにして中間言語を生成させる事で新機能を実装しやすくさせている。この事によって Jeliot2000 では教える事が困難だったオブジェクト指向を提示しやすくなった [4–6]。現在、Jeliot3 は Java 以外の言語でもアニメーション化する事や静的変数、多次元配列をサポートする事、テキストや音声などによる解説をする事、ユーザーのスキルレベルに応じたアニメーションの改良などのプロジェクトが進んでいる [5]。プログラミング学習に対して Jeliot3 を使ったの学習効果はあり、特に初学者に対する効果は研究によって実証されている [6, 7]。

2.3 Jeliot3 の問題点

Jeliot3 はアニメーションを行う時にソースコードを用いて現在行われている動作を表示する。しかし、ソースコードが記載されているエディター部分とアニメーションが行われているシアター部分に対して学習者は両方に集中しなければならないので、動的にプログラムの動作を確認しようとすると学習者の視点が一点に集中できない。また、条件分岐やループ文ではプログラムが実行している階層がソースコードでしか読み取れない問題がある。図 **4** では for ループの条件判別を行っている。Jeliot3 では条件判別により動作が変更される時はこの様に文章で分岐する事を伝えている。また、条件分岐により中括弧の中を実行するが、図 **5** や 図 **6** のようにアニメーションでは実行している階層を読み取ることができない。それに加え、次に実行する場所を示すメッセージの表示時間が短く、一時停止や表示の調整をしない限りすぐに消えてしまうので学習者が困惑する。これではソースコードを読み慣れていない初学者が実際にプログラムが実行している部分を把握することが難しくなる。

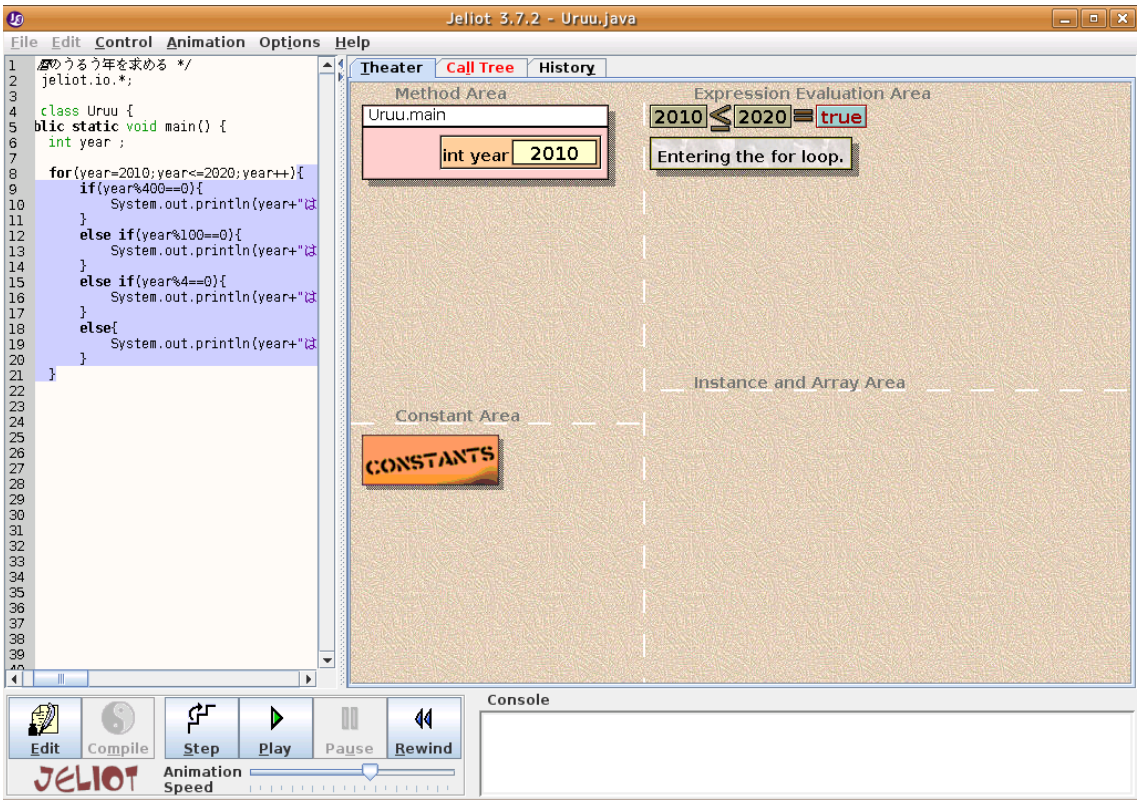

図 4　Jeliot3 の for ループ例

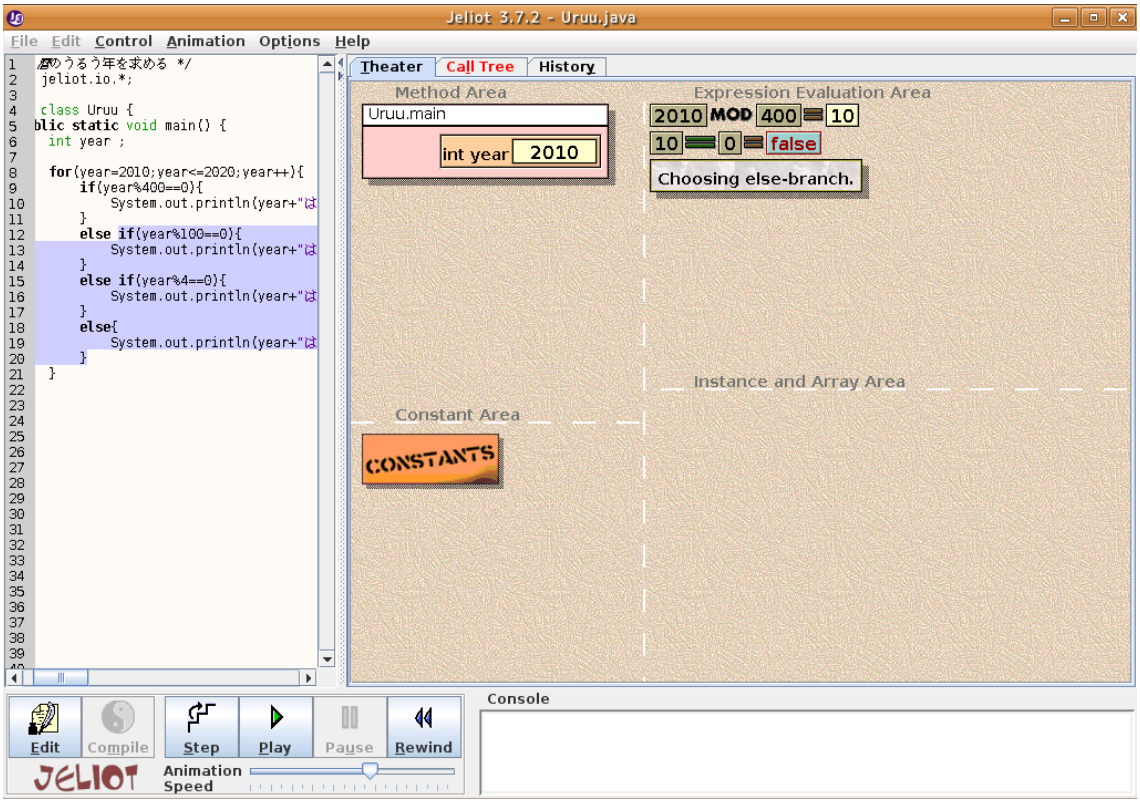

図 5　Jeliot3 のループ内での条件判定例

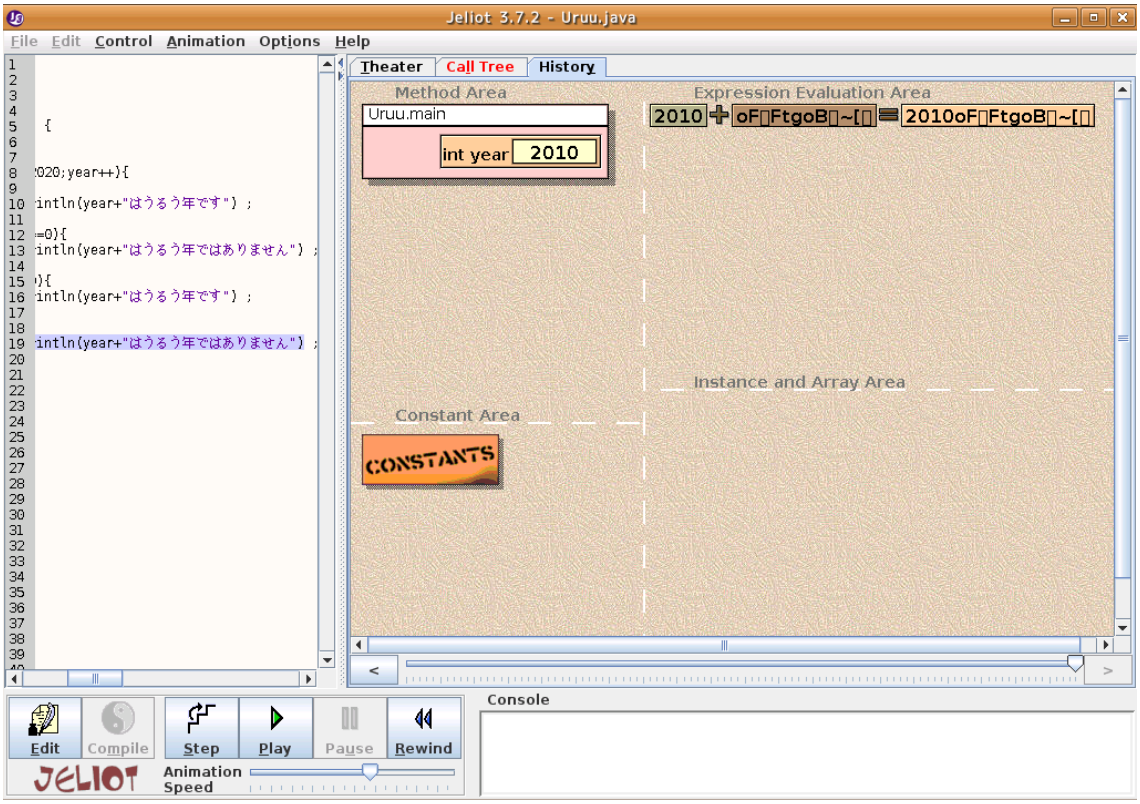

図 6　Jeliot3 の if 文で条件一致した部分の中の実行例

2.4　本研究の目的

本研究の目的は先行研究から Jeliot3 のアニメーションに着目し、アニメーションのみでプログラムの動作を説明する事でシアター部分のみに視点を集中させ、学習者の負担を軽減させる事である。また、条件分岐やループの時に階層を持たせる事により、プログラムが動作している場所をわかりやすくして学習者が困惑することを防ぐ。

3 方法

3.1 解決案

Jeliot3 が持っている問題を解決する方法としてアニメーション部分に情報を集約するシステム案を提案した。このシステムでは Jeliot3 をベースにアニメーション実行部分のソースコード部分をハイライトするだけではなくシアター部分に実行している部分を表示させ、学習者の視点の移動をできる限り抑えるアニメーションにした。また、条件分岐やループを用いた時、階層に応じてウインドウを作成しプログラムの実行している階層を把握する事ができる。このシステムによって利用者の負担を減らし、効率的に学習することが期待できる。

3.2 システム概要

システムのアニメーションを実行するとシアター内に可視化した情報が表示される。その時、現在実行している実行文をシアター内に表示する。条件文やループの時は条件部分から現在行っている部分だけを取り出して表示する。Jeliot3 では静的な数字を使う場合は図 **7** のように CONSTANTS エリアに代入に使う静的な数を作成する。その後、図 **8** のように数字を利用する所に移動し、図 **9** のように代入や評価式に使っている。しかし、これでは学習者にとって CONSTANTS エリアがよくわからない未知の存在になる。

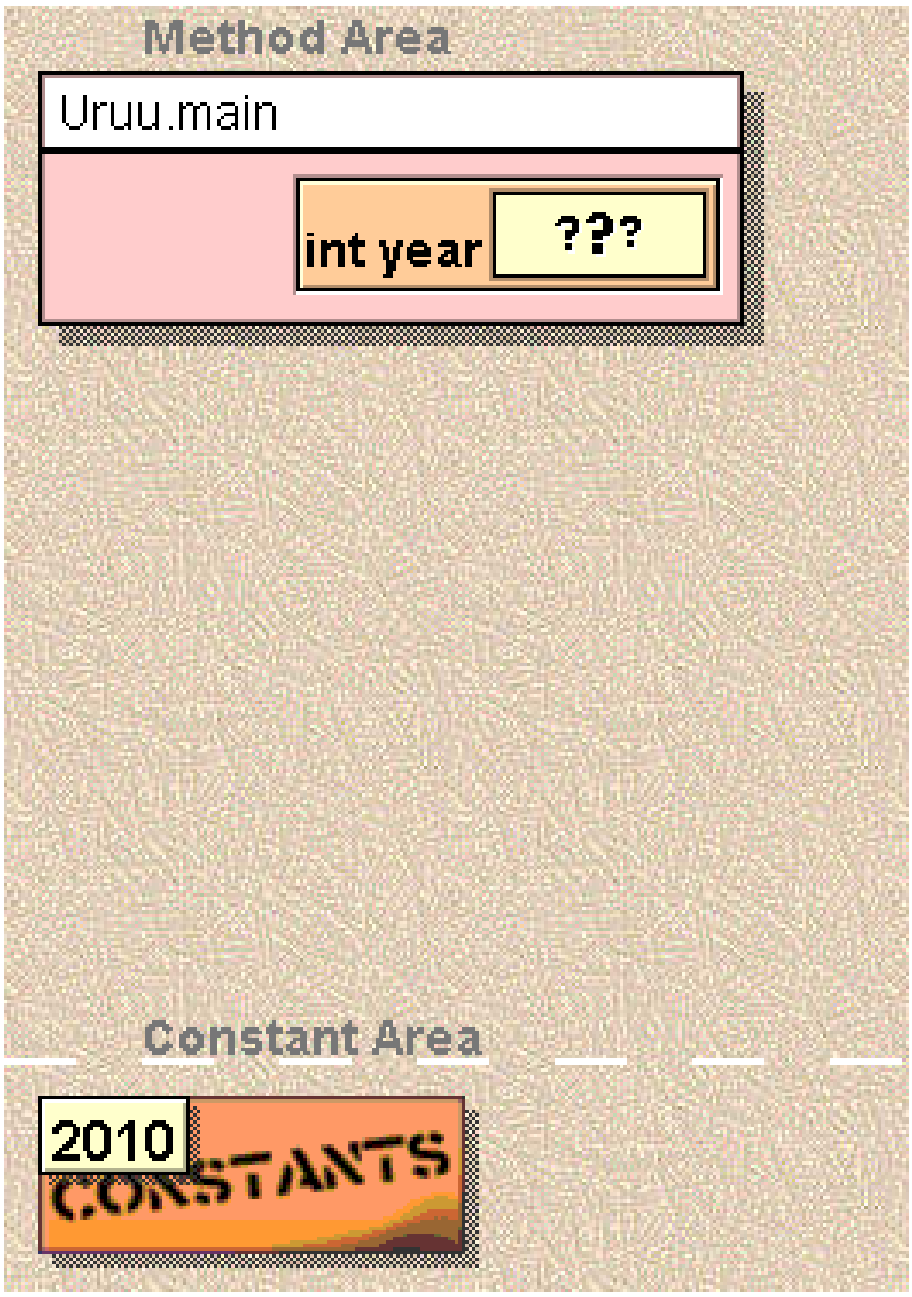

図 7　CONSTANTS エリアで静的な数、2010 を作成する

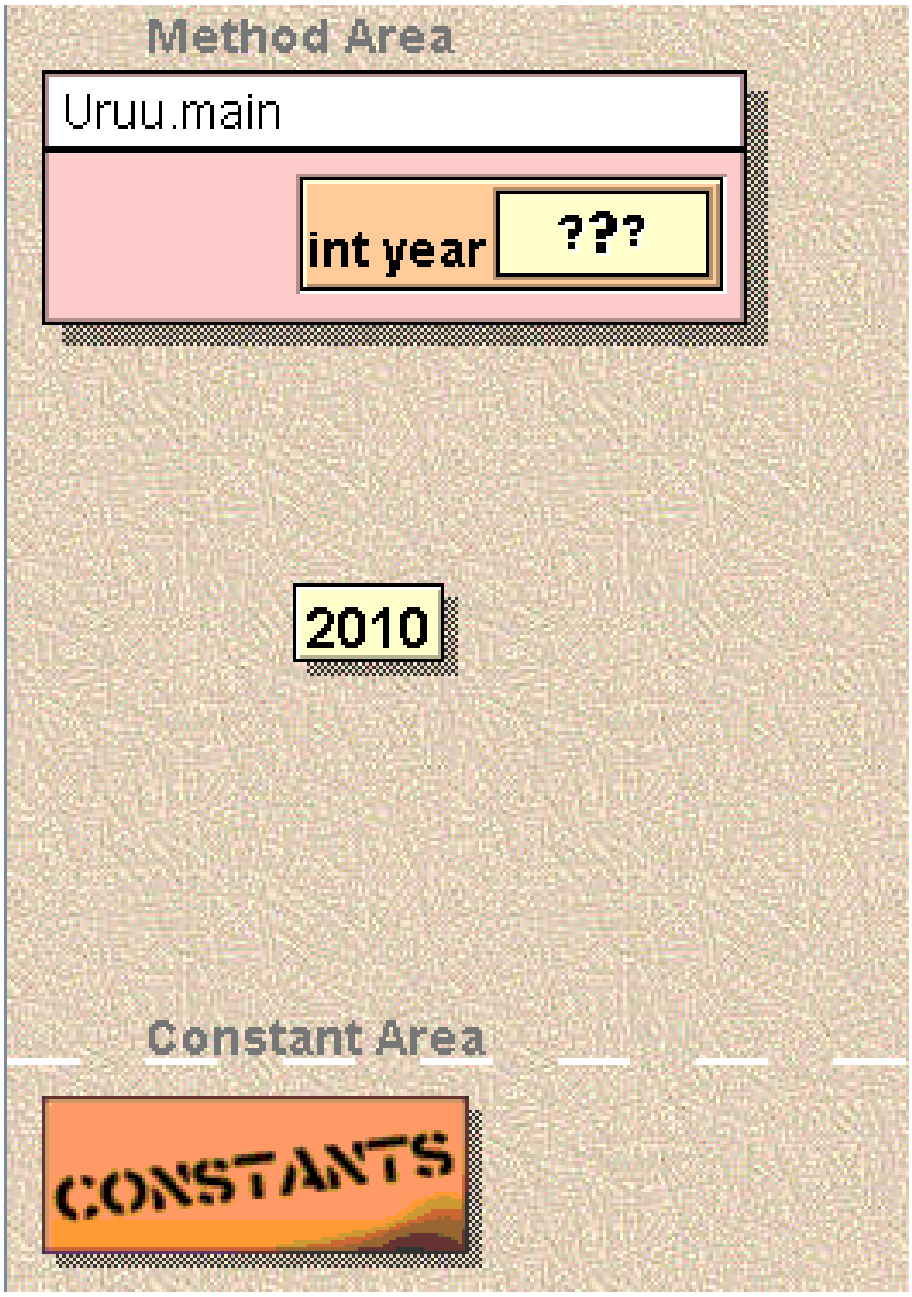

図 8　静的な数である 2010 が CONSTANTS エリアから int 型変数 year のボックスに向かって移動する

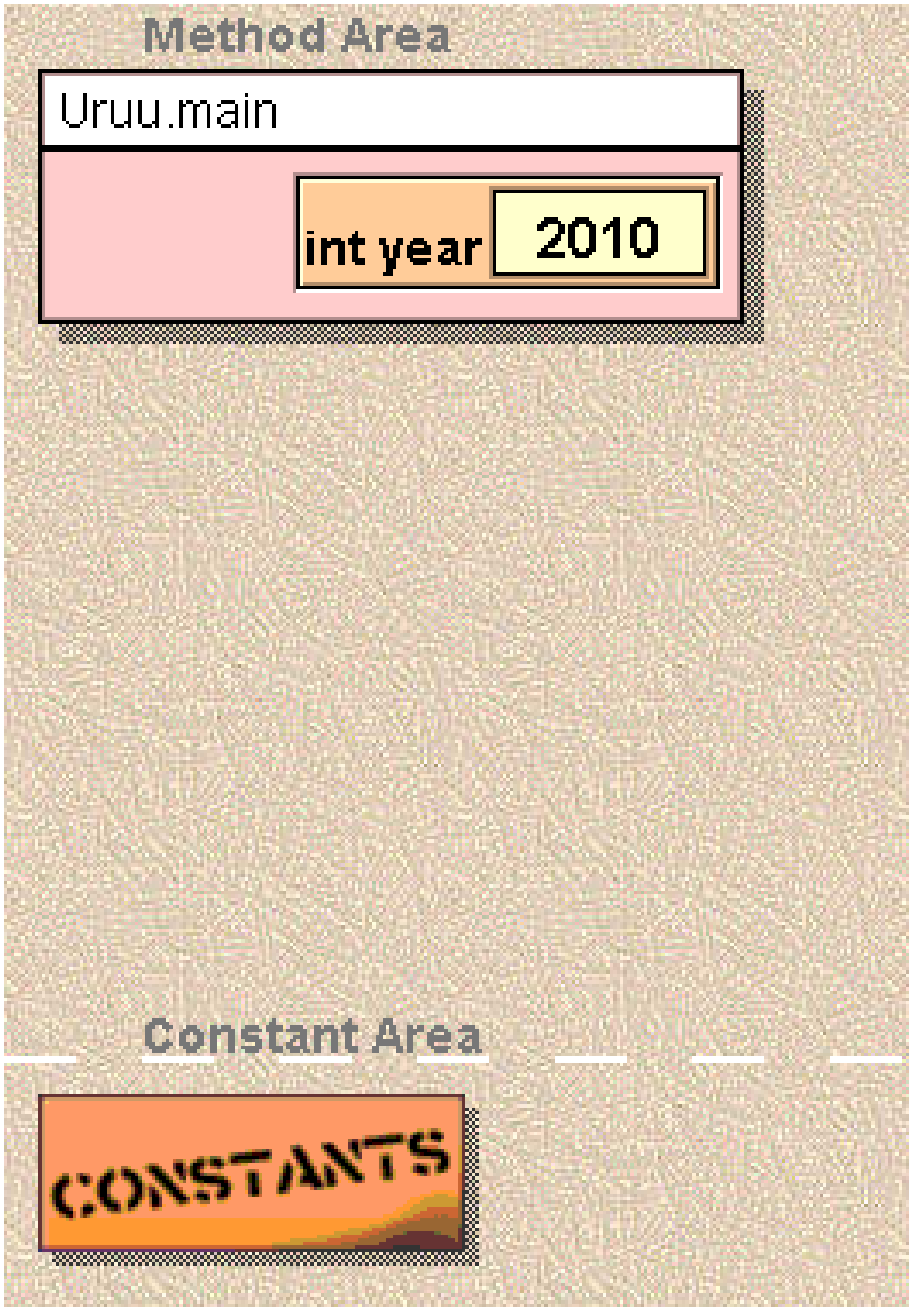

図 9　静的な数である 2010 が int 型変数 year に値が代入される

提案したシステムでは図 **10** のようにシアター内の実行文から静的な数字を作成する。作成された数字は図 **11** のようにその後、数字を利用する所に移動し、図 **12** のように代入や評価式に使う。同じように変数定義はにより指定した型を持った箱が実行文で作成される。作成された箱は定義したメソッドを示すウインドウ内に移動することによって定義する。CONSTANTS エリアを排除した事により学習者が戸惑う原因を減らした。

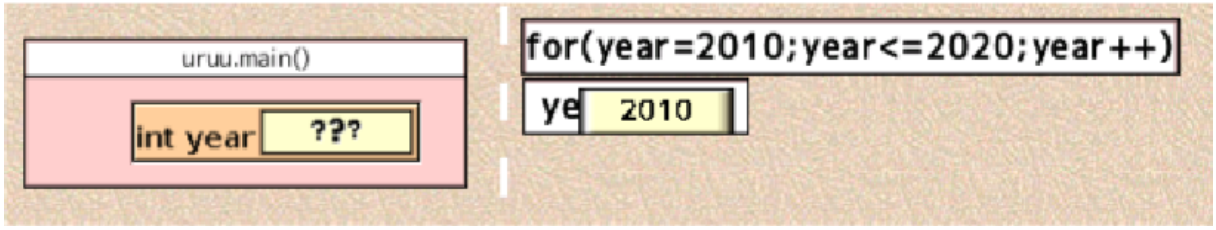

図 10　シアター内で提示したコードから静的な数、2010 を作成

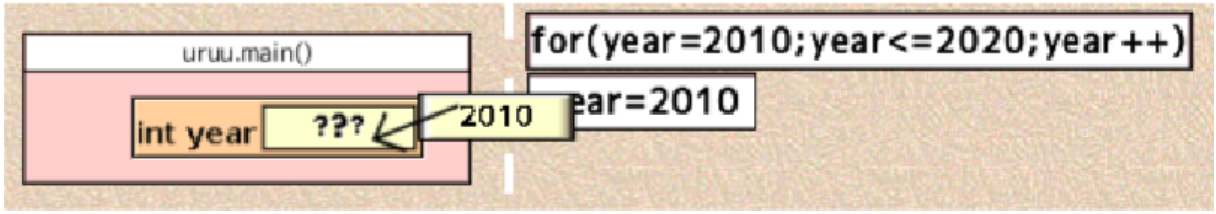

図 11　静的な数である 2010 がシアター内で提示したコードから int 型変数 year のボックスに向かって移動する

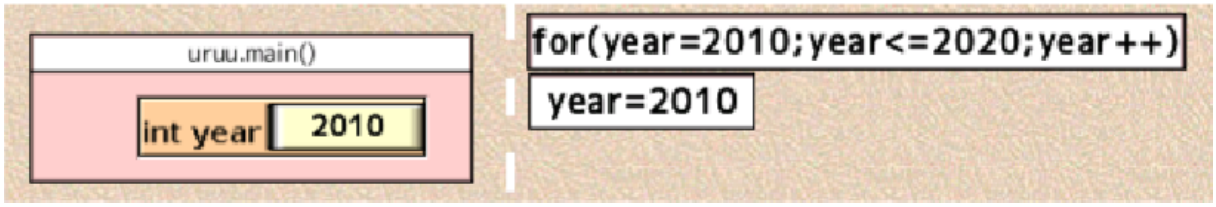

図 12　静的な数である 2010 が int 型変数 year に値が代入される

for ループはまず図 **13** のように for ループのコードをシアターに提示する。その後、図 **14** のようにループのコードから第一文のコードのみを表示したウインドウを作成する。それを図 **15** のようにループのコードの下に移動させる。移動後は第一文のコードを実行する。ループにおけるコード提示と実行のしかたはすべてこの方法で実装する。

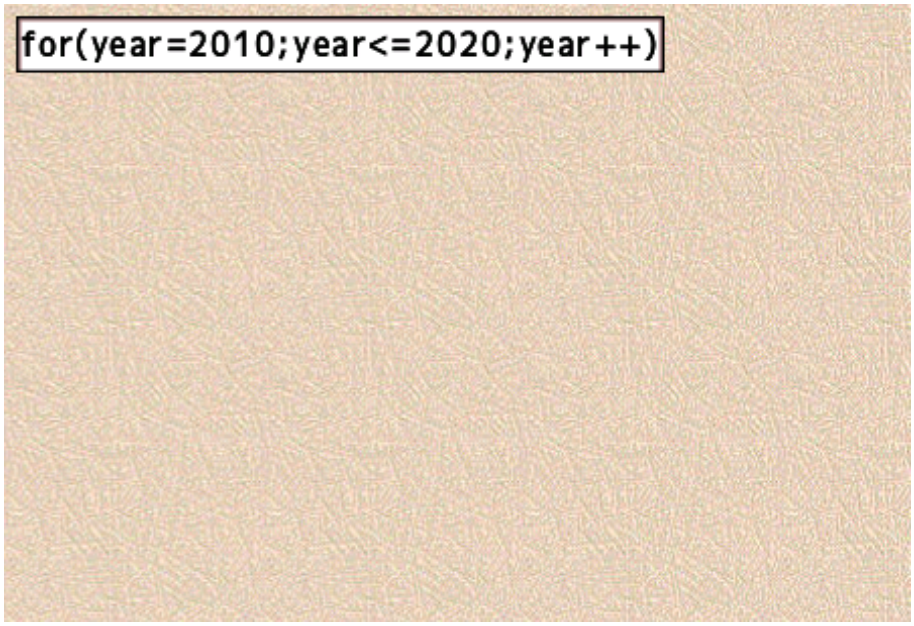

図 13　for ループのコードを表示

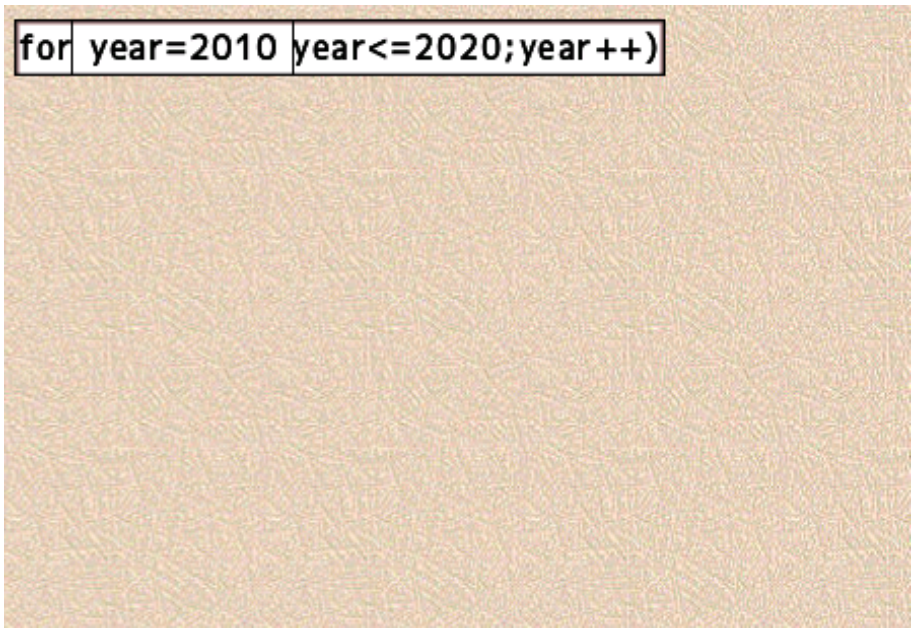

図 14　for 文のコードから第一文のコードを別で表示する

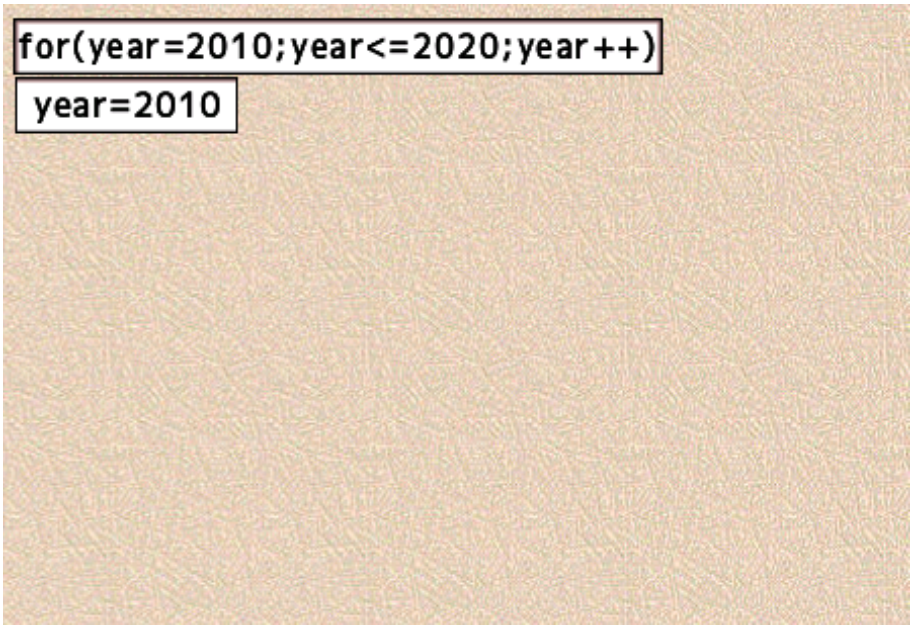

図 15　作成したコードを for 文のコードの下に移動させた後、第一文のコードを実行する

次に for ループの第二文のコードを図 **16** のように表示する。その後、図 **17** のように式を評価する。第二文の評価が true の場合、図 **18** のように for ループの中身を表すウインドウを条件文の下に作成する。第二文の評価が false になり、ループを終えた時はウインドウを消去する。while ループの場合、for ループの第二文以降の動作と同様である。

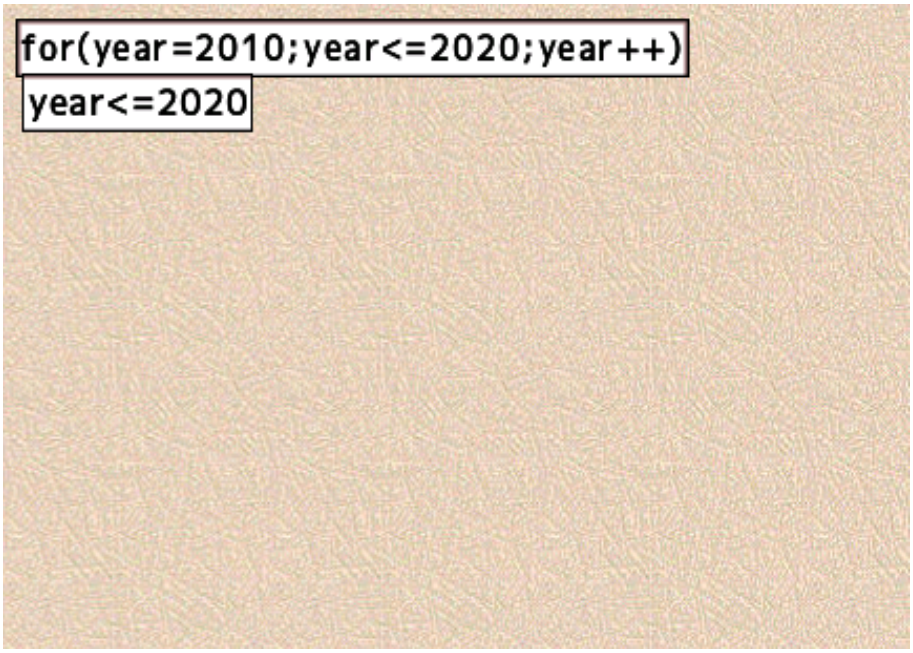

図 16　for 文のコードから第二文のコードを表示する

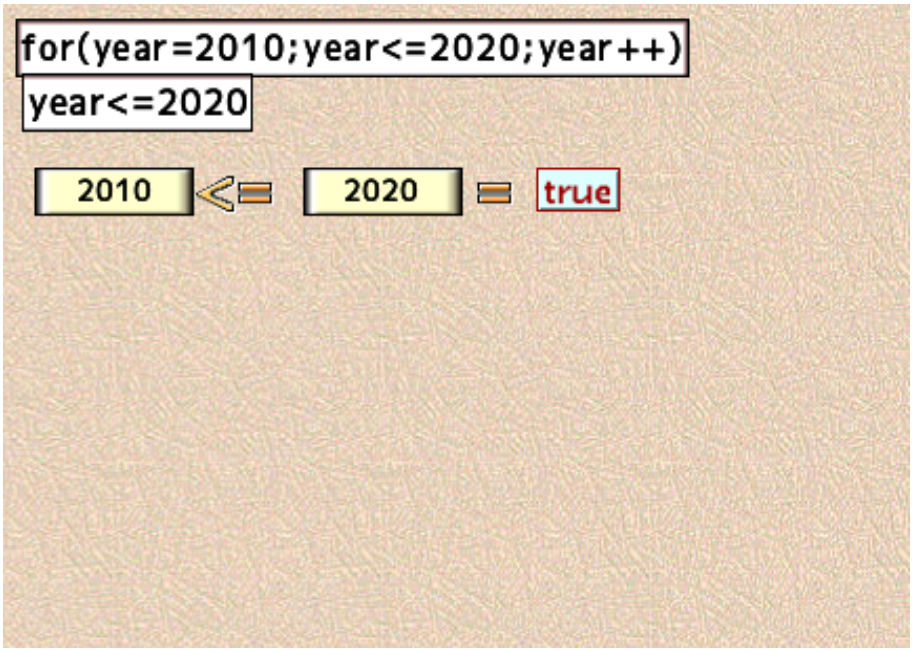

図 17　第二文の式を評価する

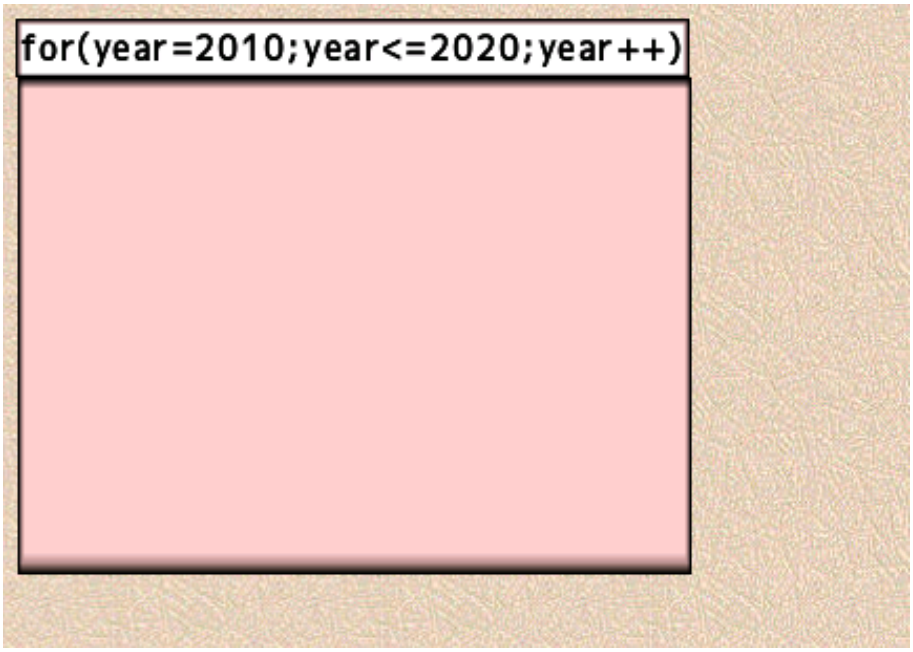

図 18　第二文の式が true であれば for で実行するコードを表示するウインドウを作成

if 文を使った時のアニメーションは図 **19** のように if から else if、else までを一つのブロックとして扱い、ブロック内のすべての条件を表示する。その後、条件評価をするためのウインドウを作成し、図 **20** のように条件を展開する。そして、図 **21** のように式を評価する。条件が一致しなかった時、図 **22** のように未評価の式を上にスライドさせる。

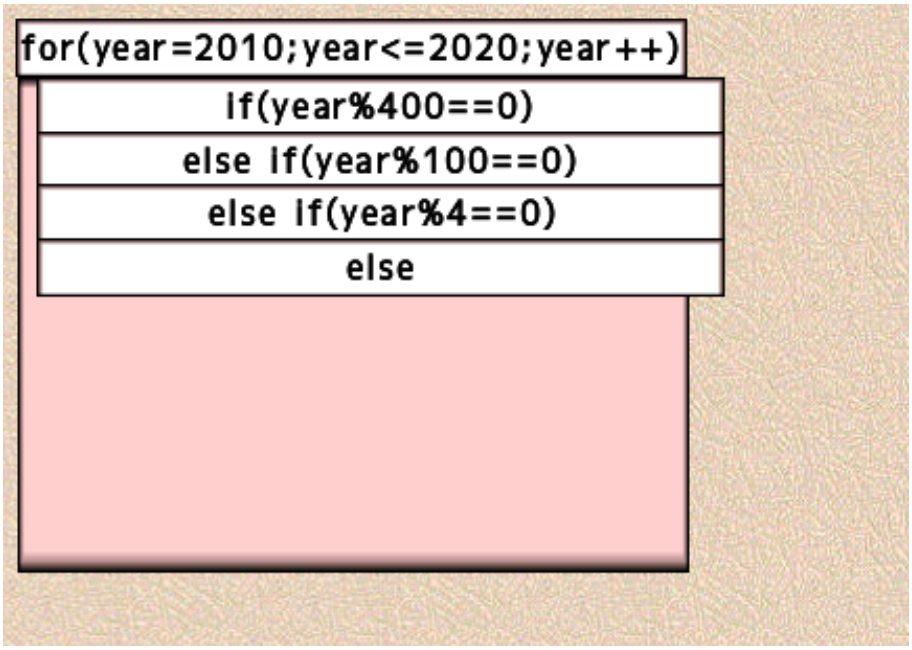

図 19　if から else if、else までをブロックとして扱い、ブロック内すべての条件式を表示

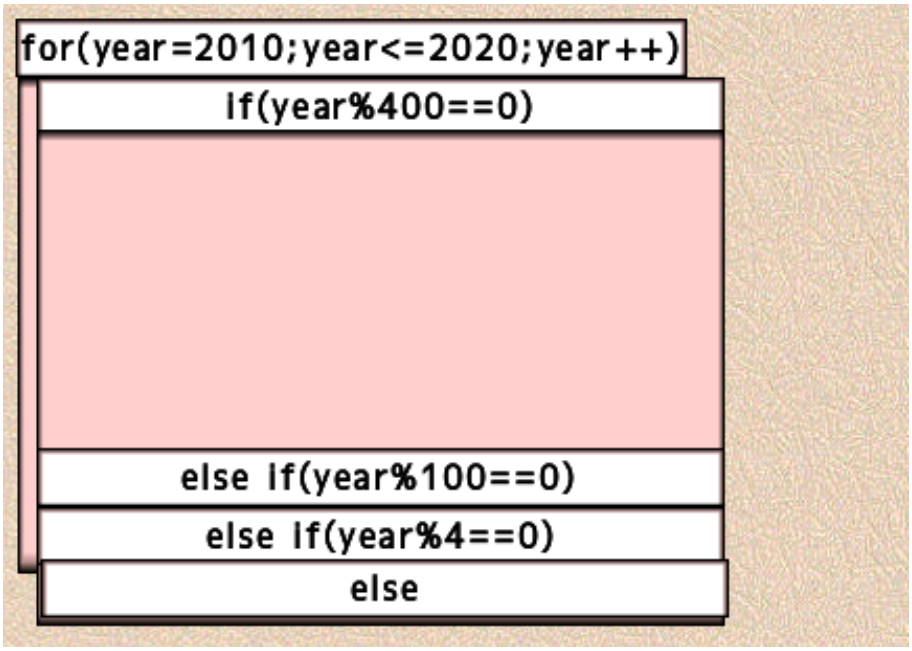

図 20　評価を行う式を上に残し、それ以外を下にスライドさせて if 文の評価を行うウインドウを展開

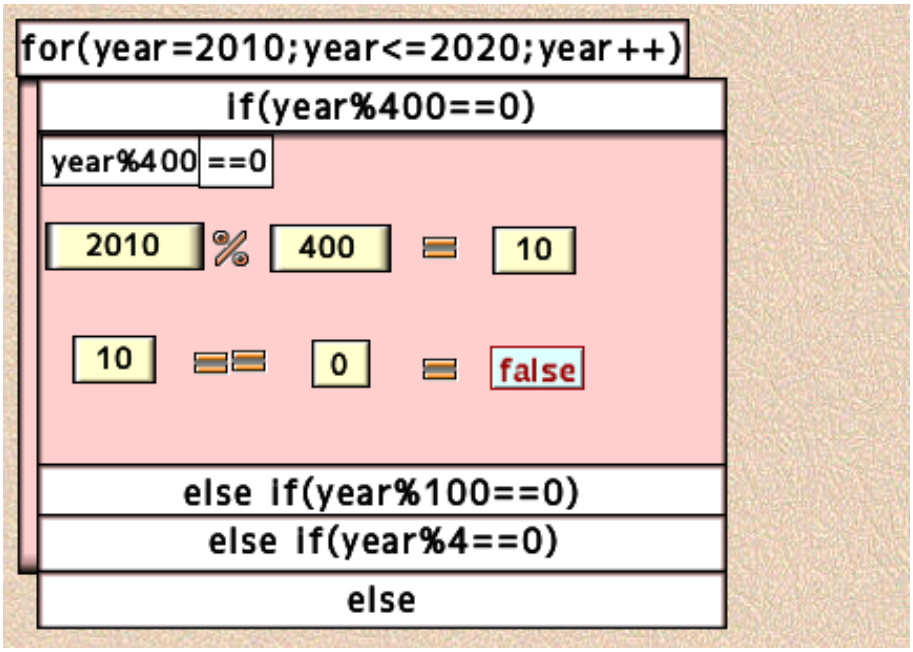

図 21　展開したウインドウ内で if 文の式の評価を行う

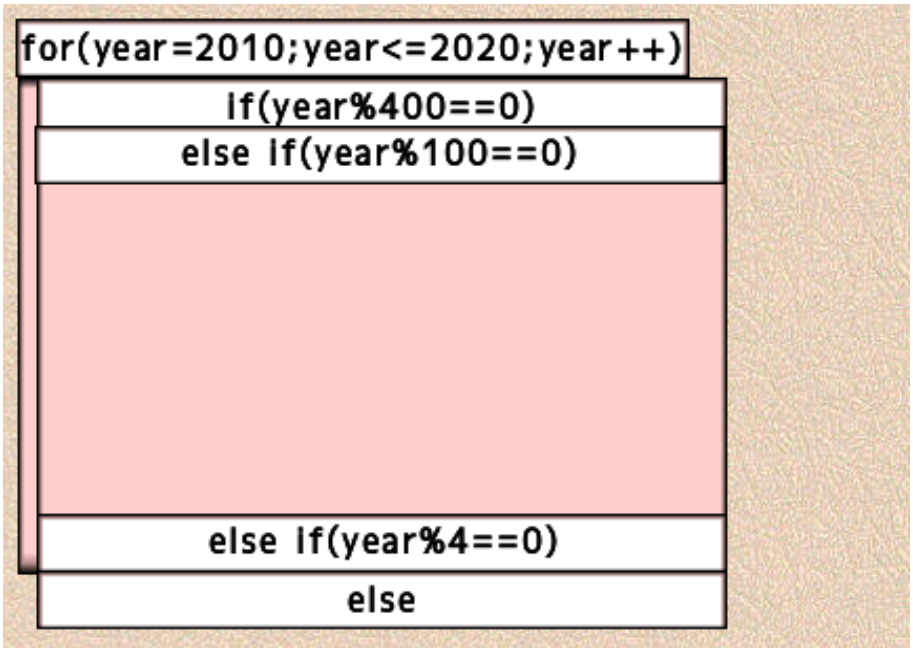

図 22　式の評価が false の場合、下にある条件文の上部の式が次に評価する式として上に移動する

条件が一致した文もしくは else の時は図 **23** のように文を起点にして中の階層を表すウインドウを作成する。そのウインドウ内で図 **24** のように一致した条件文の中のコードを実行していく。中の実行文が終了すると図 **25** のようにすべてのウインドウを消去する。

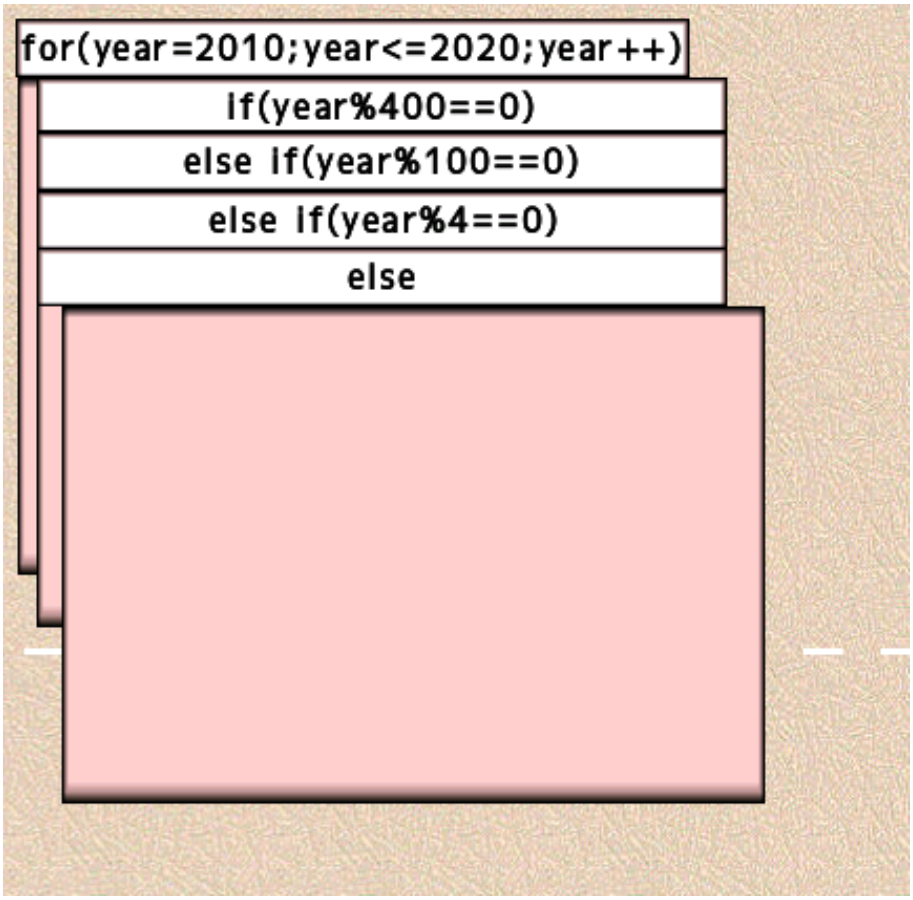

図 23　if の条件が true もしくは else の時、中で実行するコードを表示するウインドウを作成

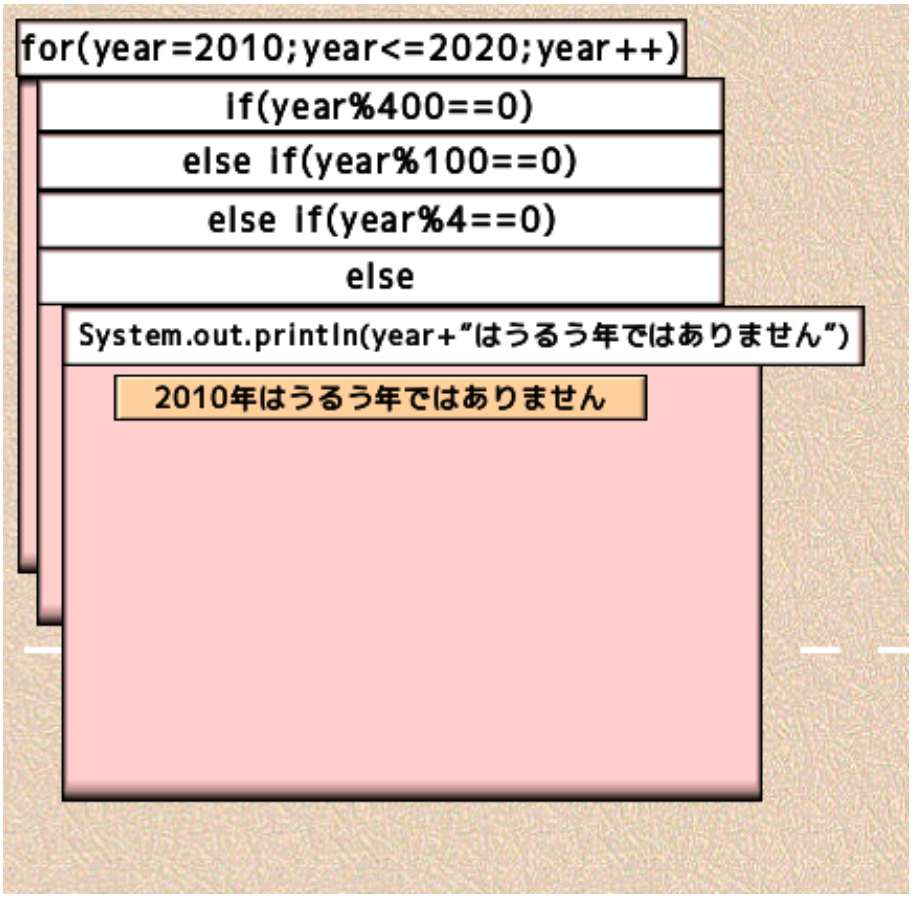

図 24　作成したウインドウ内で条件文の中のコードを実行する

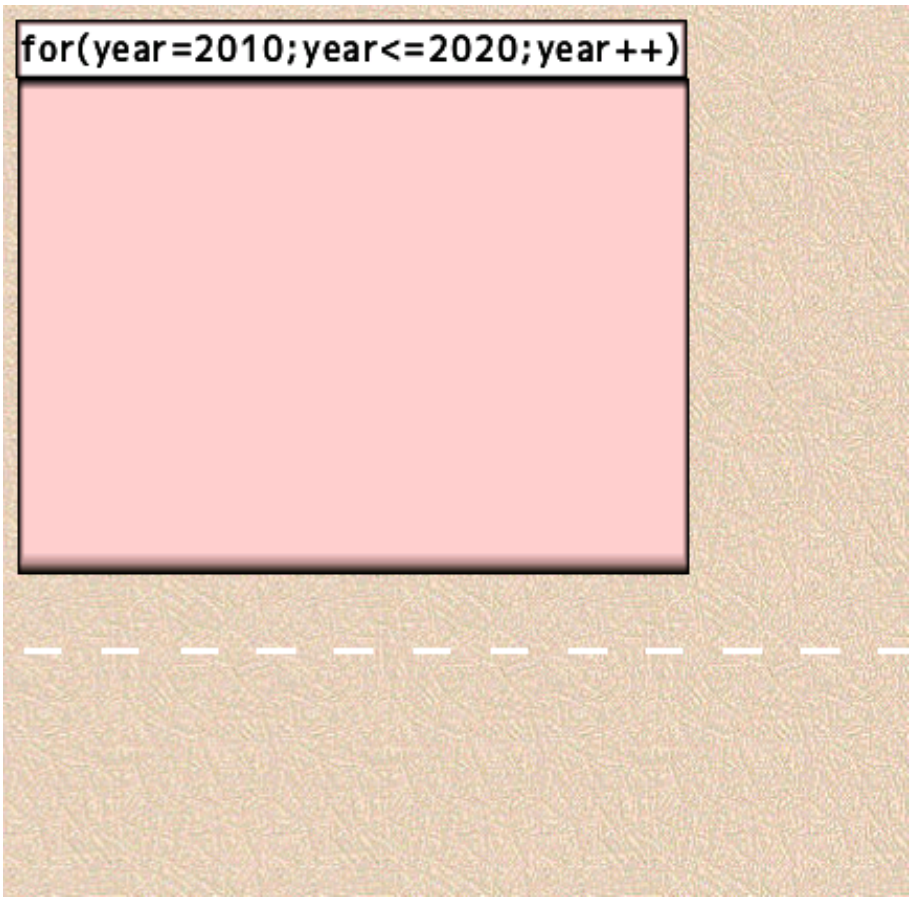

図 25　ブロック内の実行コードがすべて終えると、条件分岐で作成したウインドウを消去する

4 考察

4.1 予想される効果

本システムではプログラムの実行内容とソースコードをすべてシアター部分に表示してあるので利用者はJeliot3 に比べアニメーションに集中する事が期待できる。また、条件分岐やループを階層で表す事でプログラムがどの部分の動作を行っているかをアニメーションで理解することができる。それによって学習者は必要な情報を得やすくなり、学習時の負担を軽減する事ができる。この事から Jeliot3 が持っている学習効果と合わせると本システムはプログラムの流れの把握が容易になり、学習時に困惑する事が少なくなる。結果、プログラムの動作がより理解しやすくなり、そこからアルゴリズムやプログラムの構造を学習することができる。また、プログラムの動作とソースコードの内容を照らし合わせることにより理解が深まる [2, 8]。その結果、プログラムの動作をイメージしながらプログラミングが行うことができ、プログラミングの理解ができるようになると期待できる。

4.2 今後の展望

提案したシステムを Jeliot3 の API を使って作る事。Jeliot3 の Web ページでソースコードが公開されているので容易に入手する事ができる。Jeliot3 のビルドは主に Apache Ant を使って行う。システムが生成できたら学生に試す事。実際にプログラミング初学者にシステムを利用してもらい、提案したシステムが効果があるかを確認する。そのデータから統計を取り、効果を実証する。

5　まとめ

提案したシステムのアニメーションの動作の仕方はプレゼンテーションソフトウェアで作成したので Java 及び Jeliot3API の仕様を研究してシステムを構築してプログラミング初学者に対して実験する事である。また、Jeliot3 は本研究で提示した問題点以外にシステムの拡張性や教育者が利用するためのサポート、Java 以外の言語に対応などの問題がある [5]。プログラミング導入教育を行う上でプログラムの可視化だけではなく、学習者や指導者が利用しやすいシステムの提案も行う必要性がある。

謝辞

本研究を行うに当たり、御指導及び御協力をしていただきました大垣 斉講師には大変感謝します。また研究室のメンバー、卒業した先輩方には大変お世話になりました。

参考文献

[1] 王文涌 池田満 李峰栄「プログラミング教育における動機づけ教授方法の提案と評価」,2007 日本教育工学会論文誌 31(3), 349-357, 2007-12-20

[2] 黒河優介 三浦悠太 藤枝崇史 清水智公 服部隆志 萩野達也「内部データの視覚化によるプログラミング支援ツール」, 情報処理学会第 70 回全国大会講演論文集 ”1-307”-”1-308”, 2008-03-13

[3] 森下 博「学習意欲向上を目指したプログラミング授業の実践的展開」, 社団法人 私立大学情報教育協会 平成 20 年度 全国大学 IT 活用教育方法研究発表会 D-8

[4] Andres Moreno and Niko Myller “PRODUCING AN EDUCATIONALLY EFFECTIVE AND USABLE TOOL FOR LEARNING, THE CASE OF JELIOT FAMILY”,University of Joensuu 2003,In Proceedings of International Conference on Networked e-learning for European Universities (CD-ROM). Granada, Spain: EUROPACE

[5] Niko Myller “The Fundamental Design Issues of Jeliot 3”,Master’s Thesis, Department of Computer Science, University of Joensuu 20.2.2004, (未公刊)

[6] Kimmo Sivula “A Qualitative Case Study on the Use of Jeliot 3”,University of Joensuu Department of Computer Science Master’s thesis 26.8.2005, (未公刊)

[7] Mordechai (Moti) Ben-Ari “The Effect Of The Jeliot Animation System On Learning Elementary Programming”,Department of Science Teaching Weizmann Institute of Science, 4th Panhellenic Conference on Didactics of Informatics, Invited speakers

[8] 今泉俊幸 橋浦弘明 松浦佐江子 古宮誠一「ブロック構造の可視化環境によるプログラミング学習支援」, 信学技報，vol.109 no.193 ET2009-30 pp.45-50,2009 年 9 月