WebGL とジェスチャー認識デバイスを用いたプログラミング例題提示システムの開発

Development of a Program Visualization System using WebGL and Motion Sensing Input Devices

森田昌樹[*1], 香川考司[*2]
Masaki MORITA[*1], Koji KAGAWA[*2]
[*1] 香川大学工学研究科
[*1]Graduate School of Engineering, Kagawa University
[*2] 香川大学工学部
Faculty of Engineering, Kagawa University
Email: s14g484@stmail.eng.kagawa-u.ac.jp

あらまし：WebGL を用いた 3D グラフィックによるプログラミング初心者向けの学習支援システムと Kinect または Leap Motion を用いた NUI (Natural User Interface) を用いた入力システムを開発する。
キーワード：WebGL, Kinect, OpenNI, Leap Motion, NUI

1. 背景

プログラミングを学習する初心者が for 文や while 文といったループ構造などのプログラムの制御構造を理解することは難しい。よって、プログラミング初心者向けに学習支援を行うシステムを提案する。一般的な学習方法である教科書・参考書などのテキストを用いた講義形式の授業では、制御構造を理解しているか、確認することは難しい。初心者にとっては理解できないまま学習意欲が低下し、学習を放棄してしまう可能性が非常に大きい。そこで本研究では初心者にとって講義形式の授業の中でも制御構造の理解を確認しながら学習できる教材としてのシステムを開発する。

まず初心者である学習者に楽しく学習してもらうにはどのような方法を取るのが良いのかを考えた。テキスト形式の資料ではプログラムの制御構造の説明は文章またはフローチャートの場合がほとんどであり、学習者側は文章を読んだだけでは理解が難しい場合が多い。そのため、制御の流れを分かりやすくするために 3D グラフィックスを用いて可視化を行う。さらに Kinect や Leap Motion などのジェスチャー認識デバイスによる操作に対応させることで、ゲーム感覚で楽しく学習してもらうことを目的としている。

ジェスチャー認識デバイスは上下奥行きなどの 3 次元の情報の取得が可能であるので 3D グラフィックスと相性が良いと考えた。手の動きやジェスチャーを用いて、3D 空間でのカメラやプレイヤーの操作を想定している。

3D の表示部分では、Web ブラウザーを利用した 3D グラフィックスの規格である WebGL を用いてこのプログラムの構造を 3D オブジェクトとして可視化する。3D にすることで学習者が直感的に見て理解しやすい例題提示システムを提供する。

Kinect から取得したデータを OpenNI (Open Natural Interaction) ライブラリにより制御し、サーバー上の JavaServlet を通してブラウザー上で扱える形に変換するインタフェース部分のシステムが既存の研究では完成されている。Han らの研究[(1)]ではソースプログラムを構文解析して中間言語に変換し、この中間言語を 3D 可視化部で処理して 3D モデルを作成する Web システムが開発された。このシステムは制御構造の振舞いを示しているわけではない点とユーザーからのインタラクションが未実装な点があり、本研究ではこれらを参考に 3D 空間生成部分を実装し、初心者を対象とした分かりやすいプログラミング学習支援システムを開発することを目的とする。このシステムは演習への導入としてプログラムを理解してもらうための利用や、教員が講義中に教材としてスクリーンに映し出し、デモとしての利用を想定している。

代表的な可視化システムの例として UUhistle[(2)]と Jeliot 3[(3)]を参考にした。これらのシステムは 2D のアニメーションでプログラムの動作を可視化しているが、プログラミング初心者にとっては直感的に分かりづらく、アニメーションを目で追いにくい点などが挙げられた。これらの点に注意しシステムの開発を進めた。

2. システム概要

システムは学習者にとって導入作業が不要である Web ベースシステムとする。他の Web システムとの連携が容易な点や、教員にとってシステム導入の指導を行わなくてよい点が大きなメリットである。

本システムは Kinect または Leap Motion からの入力操作部分、ソースコードから条件式の位置を探し出す解析プログラム、制御構造解析プログラムから得られたデータを元にソースコードを 3D で可視化し、ブラウザーで表示するプログラム部分から構成されている。

本システムの構造は図 1 のとおりである。

3. 制御構造解析プログラム

制御構造解析プログラムとはソースコード中の while 文、for 文、if 文などの条件式の位置やインデントの深さの情報を取得するプログラムである。これにより各条件式の位置関係が把握できるようになり、可視化が容易になる。

プログラムの処理の流れとしては、初めに条件式を解析したい C 言語のソースファイルを指定して読み込む。GNU indent をもちいてソースファイルのインデントを整える。次に、ソースコードを 1 行ずつ読み込み、先頭から半角スペースの字数でインデントの深さを調べる。そして if 文や for 文などの条件式が出現すると、データ保存用の配列に「条件式が出現したインデントの字数」、「条件式の種類」、「条件式が出現した行数」を格納する。また、閉じブレースが出現するとインデントの字数を前述の「条件式が出現したインデントの字数」と照らし合わせ、同じであれば、「条件式が終了する行数」としてカウントし配列に格納する。これにより各条件式の位置関係が把握できる情報が取得することが可能である。

4. WebGL による 3D 描画処理とプログラムの可視化

ウェブブラウザーで 3D グラフィックスを表示させるための仕様は WebGL を利用する。WebGL の補助ライブラリとして Three.js を用いる。

まず、3D オブジェクトを描画するためにはシーン (scene) と呼ばれるオブジェクトを配置するフィールドを生成する。このシーンの中にオブジェクトやそれを照らす光源、さらにそれを映し出すカメラなどの 3D の描画に必要な要素を追加する。そして制御構造解析プログラムより解析結果を配列として受け取り、条件式の位置や種類を把握し、種類に応じたオブジェクトを位置情報に従い配置する。図 2 は WebGL を利用した 3D グラフィックスでプログラムの制御構造を可視化した例である。立方体のオブジェクトは Three.js の CubeGeometry メソッドを用いて作成し、座標を指定してシーンに追加する。立方体を結ぶ直線は始点座標と終点座標を指定し、Line メソッドを用いてシーンに追加する。アニメーションを行うためには描画処理を行う関数 render を定義し、render 関数内でシーンとカメラの位置情報に変更があった場合は変更を反映して再描画を行うメソッドを利用する。さらに render 関数自体は requestAnimationFrame メソッドを用いて毎秒約 60 フレームで繰り返し実行する。

5. まとめ

プログラミング初心者への教育支援のため、WebGL とジェスチャー認識デバイスを用いたプログラミング例題提示システムの提案と構築を行った。

Web ベースのシステムである点に関して、システム全体については未完成であるが、目標としていたシステムの実現が可能な段階までは調査を進めた。Kinect または Leap Motion からのデータをブラウザー上で扱える形に変換する方法は確立済みであり、あとはそのデータをインタラクティブ 3D アニメーションに反映させるだけである。可視化したいソースコードを解析するために制御構造解析プログラムを作成し、その解析結果を反映して 3D オブジェクトの配置を行う部分は実装済みである。

今後の課題としては、3D アニメーション、本システムをサーバー上に配置しインストールを必要とせず、ブラウザーでアクセスするだけで利用できるようにシステムを構築するなどが挙げられる。

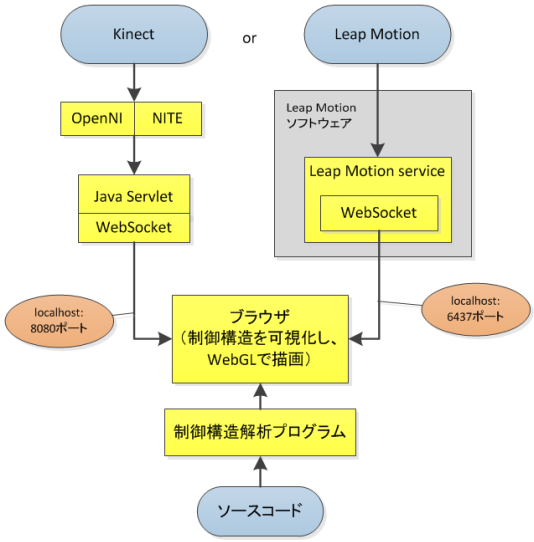

図 1：システムの全体構造

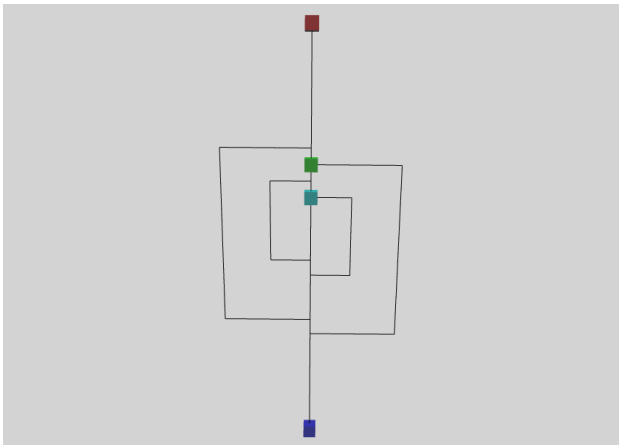

図 2：システムのスクリーンショット

参考文献

(1) Han and Kagawa, “Towards a Web-based Program Visualization System using Web3D” ITHET 2012, 2012.
(2) Sora and Sirkia, “UUhistle: a Software Tool for Visual Program Simulation,” Proceedings of the 10th Koli Calling International Conference on Computing Education Research, Koli Calling '10, pp. 49 - 54, 2010.
(3) Moreno, 他 , “Visualizing Programs with Jeliot 3”, Advanced Visual Interfaces AVI 2004, pp. 373 - 376, 2004.