

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

ビジュアルプログラミング言語による ビックデータ解析システムの開発

Development of Big Data Analysis System
Using Visual Programming Language

Keniti Numata

Graduate School of Information Engineering, Toyama Prefectural University
u055017@st.pu-toyama.ac.jp

Teams, 9:50-10:15 Friday., December 4, 2020,
Toyama Prefectural University.

1.1 本研究の背景

2/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

背景

近年、企業などでは世間に溢れる様々な情報を収集し、ビッグデータとして様々な処理や解析によって情報を扱うことが増えている。これらのデータは、膨大であるため一般的にプログラミング言語を使って処理される。

ビッグデータについて

- ① ビッグデータとは、大量で高頻度な多様性があるデータとして定義される。
- ② ビッグデータの例として、ログデータや顧客データ、購買データなどがある。
- ③ 需要の予測やコストの削減、仕入れなどの最適化など様々な活用方法がある。

1.2 本研究の目的

3/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

現在の問題点

① ユーザビリティの低さ

プログラミング初心者にとって、プログラミングは手にかけづらい。

② データの取り扱い

ビッグデータを簡単に解析する手法が存在しない。

本研究の目的

- ・ プログラミング初心者でも扱いやすい環境の開発
- ・ ビッグデータを処理できるように開発
- ・ 外部に公開し、複数人から利用できるようにする。

2.1 ビジュアルプログラミング言語

4/19

ビジュアルプログラミング言語

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

プログラムをテキストで記述するのではなく、視覚的なオブジェクトで記述するプログラミング言語のこと。視覚的でわかりやすいものが多いいため、プログラムの組み立て方を学ぶのに有効であると注目されている。

ビジュアル言語

ブロックタイプ



テキスト言語の論理に近い

例 Scratch・MakeCode...

フロータイプ



フローチャート的

例 MESHアプリ...

独自ルールタイプ



独自の考え方

例 Viscuit...

2.1 ビジュアルプログラミング言語

5/19

ブロックタイプのビジュアルプログラミング言語

機械学習（人工知能・AI）を使って課題を解決するクラウドサービスの MAGELLAN BLOCKS（BLOCKS）や教育用作られ様々なアプリケーションに応用して使われている Blockly などがある。応用して使われているサービスとして Scratch や MakeCode が存在する。

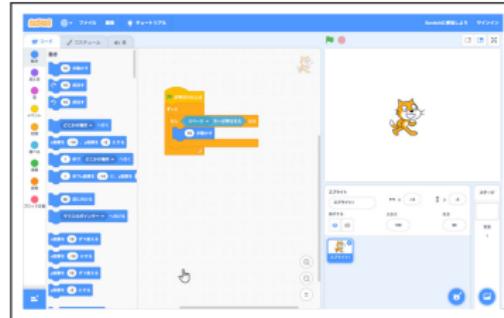


Figure 1: scratch



Figure 2: makecode

2.1 ビジュアルプログラミング言語

6/19

Blockly

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

Google が提供しているビジュアルプログラミング言語のライブラリ。簡単な記述で自分だけのビジュアルプログラミング言語を作ることができる。

また、作成したブロックは JavaScript や python, PHP, Lua, Dart などのプログラミング言語にエキスポートすることができる。また、カスタムブロックという機能があり、もともとあるブロックの他にユーザが好きなブロックを作成することができる。



The screenshot shows the Blockly interface. On the left, there's a sidebar with categories: Blocks, Logic, Loops, Math, Text, Lists, Colour, Variables, and Functions. The 'Text' category is currently selected. In the center, there's a code editor area with a 'repeat [10] times' block. Inside the loop, there's a 'do' block which contains a 'print' block with the string 'Hello'. To the right of the code editor, there are several control buttons: a play button, a stop button, a reset button, and a zoom in/out button. At the top right, there's a language selection dropdown set to 'English'.

2.2 カスタムブロック

7/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

カスタムブロックの構成

1. ブロックの定義
2. コードの生成
3. ブロックのカテゴリーと配置決め

Blockly Developer Tools

簡単にカスタムブロックを作成する支援ツールとして、Blockly Developer Tools がある。

この支援ツールは blockly を用いて、ブロックを作ることができる。

2.2 カスタムブロック

8/19

2.2.1 ブロックの定義

作成したいブロックの外観とブロックに接続する数値やテキストをここで定義する。

外観は、ブロックの色やブロックの接続(構文ブロックと値ブロック)，表示する文字等がある。

また、ブロック内の空きに何を入力(input)or 出力(output)とするか決める。



Figure 3: block の種類

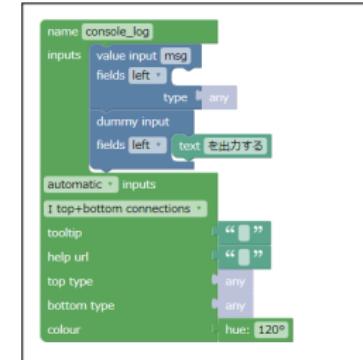


Figure 4: console log に結果を出力する関数

2.2 カスタムブロック

9/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ

2.2.2 コードの生成

コードの生成では、ブロックの動作の定義を行う。例えば、平均値を出すブロックを作成するときは、平均を出す計算部分をここで書く。

```
averagejs / ...
Blockly.JavaScript['average'] = function(block) {
    var value_v1 = Blockly.JavaScript.valueToCode(block, 'v1', Blockly.JavaScript.ORDER_ATOMIC);
    var value_v2 = Blockly.JavaScript.valueToCode(block, 'v2', Blockly.JavaScript.ORDER_ATOMIC);
    // TODO: Assemble JavaScript into code variable.
    var code = '(' + value_v1 + '+' + value_v2 + '/2';
    // TODO: Change ORDER_NONE to the correct strength.
    return [code, Blockly.JavaScript.ORDER_NONE];
};
```

2.2 カスタムブロック

10/19

2.2.3 ブロックのカテゴリーと配置決め

作ったブロックをどこのカテゴリーに入れるかを決める。

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに



The image shows the Scratch Block Library interface. On the left, there is a vertical sidebar with color-coded categories: Logic (blue), Loops (green), Math (purple), Text (teal), Lists (dark blue), Colour (brown), Variables (pink), and Functions (dark purple). Below these categories is a button labeled "Block Library". The main area displays two blocks: one green block labeled "【】を出力する" (Output []), and one purple block labeled "【】と【】の平均値" (Average of [] and []). The background of the main area has a grid pattern.

3.1 提案手法

11/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

従来の Blockly の問題点

ビックデータを扱うとき、外部からデータ (CSV) を読み込む必要があるが `blockly` のデモコードには外部からのファイルの読み込みが実装されていない。また、`blockly` はブラウザをベースにしてクライアント側で `javascript` によって処理を行うサーバ・サイド・インクルード (SSI) であるため、ビックデータの関連の処理ができるライブラリを使うことができない。

提案手法

- CSV ファイルの読み込みブロックの作成
- Ajax による非同期通信
- 機械学習ライブラリの導入

3.2 CSV ファイルの読み込みブロックの作成

12/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

ブロックの役割と見た目

実際に作ったブロックの外観と、外観作成のためのコードが以下になります。

** ファイルを選択してください ** を読み込み

```
Blockly.JavaScript['import file'] = function(block) {
    //var value_csv = Blockly.JavaScript.valueToCode(block, 'csv',
    var value_csv = Blockly.JavaScript.valueToCode(block, 'csv', Blockly.JavaScript.ORDER_ATOMIC);
    // TODO: Assemble JavaScript into code variable.
    var code = '...';
    // TODO: Change ORDER_NONE to the correct strength.
    return [code, Blockly.JavaScript.ORDER_NONE];
};
```

3.3 Ajax による非同期通信

13/19

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

非同期通信とは

POSTリクエストを出してからレスポンスが返ってくるまでの間、ブラウザ上で他の処理を行い、サーバーからのレスポンスが返ってきた時にサーバー側から所得した情報をWebブラウザ上に表示する仕組み。

Ajax 通信とは

通常、HTTPとはWebページなどをやり取りする技術（プロトコル）で、ブラウザとサーバーとの間で使われています。Ajaxを使うと、ブラウザに代わってJavaScriptがサーバーとHTTP通信することができます。

3.2 機械学習ライブラリの導入

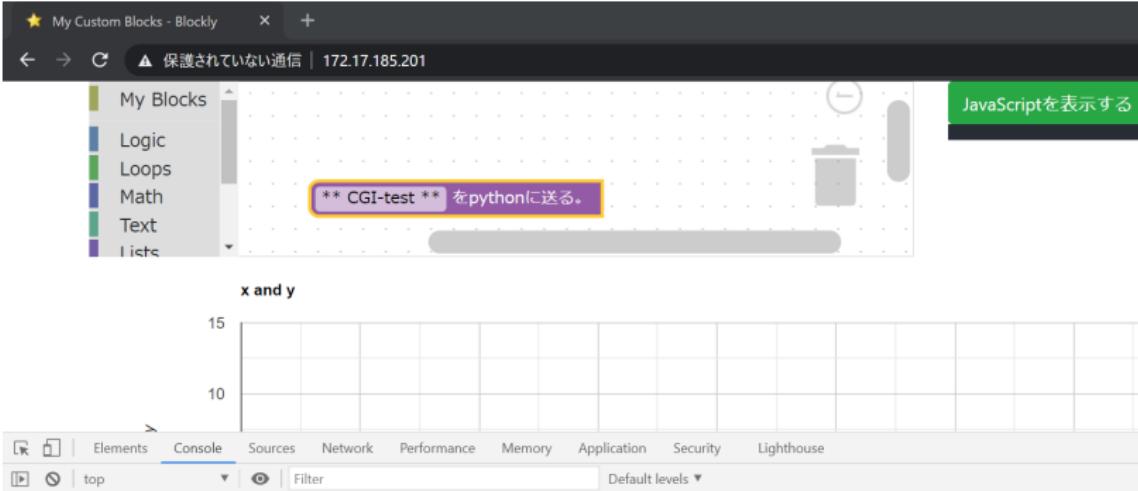
1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

4.1 数値実験

15/19

実験設定

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに



The screenshot shows a web-based Blockly editor interface. At the top, there's a toolbar with a star icon, a dropdown menu labeled "My Custom Blocks - Blockly", and a plus sign icon. Below the toolbar, a status bar displays "△ 保護されていない通信 | 172.17.185.201". On the left, a sidebar titled "My Blocks" lists categories: Logic, Loops, Math, Text, and Lists. In the main workspace, a purple block contains the text "CGI-test" followed by "をpythonに送る。". Below the workspace is a grid labeled "x and y" with axes at 10 and 15. At the bottom of the editor, there are tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Lighthouse. The "Console" tab is selected, showing the command "top" and a "Default levels" dropdown. To the right of the editor is a terminal window titled "zeta@DESKTOP-HLB5V8B:~". It shows the command "sudo service apache2 restart" being run, followed by a password prompt, the restart message, and the "[OK]" response.

4.2 数値実験結果

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

4.2 数値実験結果

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

実験結果

4.3 考察

1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに

考察

5. おわりに

19/19

まとめ

python と

今後の課題



1. はじめに
2. ビジュアルプログラミング言語
3. 提案手法とシステムのアーキテクチャ
4. 提案手法とシステムのアーキテクチャ
5. おわりに