GDG Devfest[W] Kyoto

Google Blockly で ビジュアルプログラムを 楽しむ

兼高理恵 (robo)

Google Blockly で
ビジュアルプログラムを楽しむ

自己紹介

・名前　　robo（兼高理恵）

・お仕事　　Java 技術者
設計から実装まで

・好きなもの　　モバイル端末

・補足　　Android女子部関西(リーダは日高未紗子さん)
助手兼掃除番

Blockly について

Blockly は、
Googleさんが2012/06に公開された、
ブロックのドラッグ＆ドロップのみで
任意プログラム言語のコードテキスト作成ができる、

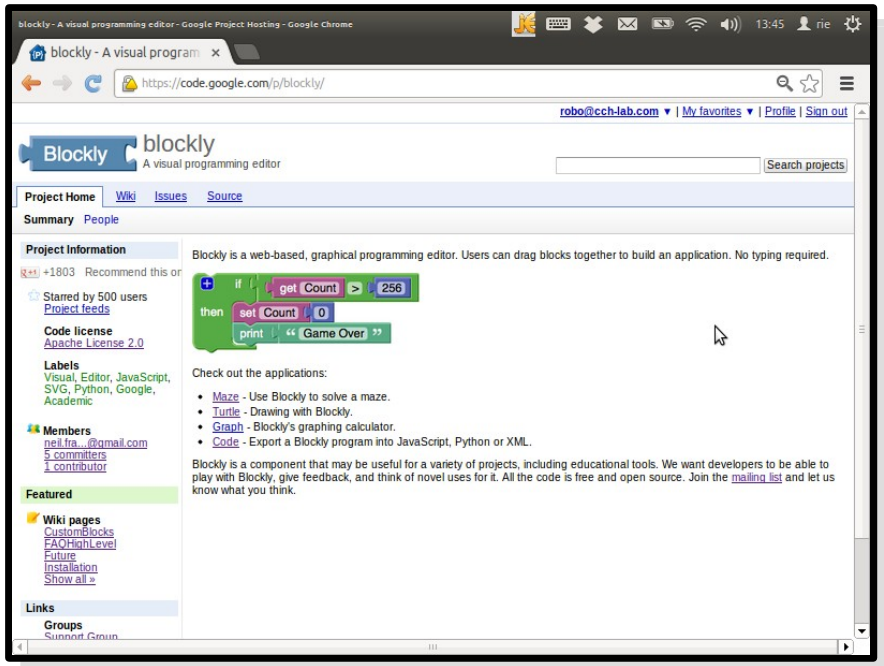
Webベースの
ビジュアル・プログラミング・エディタ環境(基盤)です。
Blockly という独自プログラム言語ではないことに
ご注意ください。

参考先

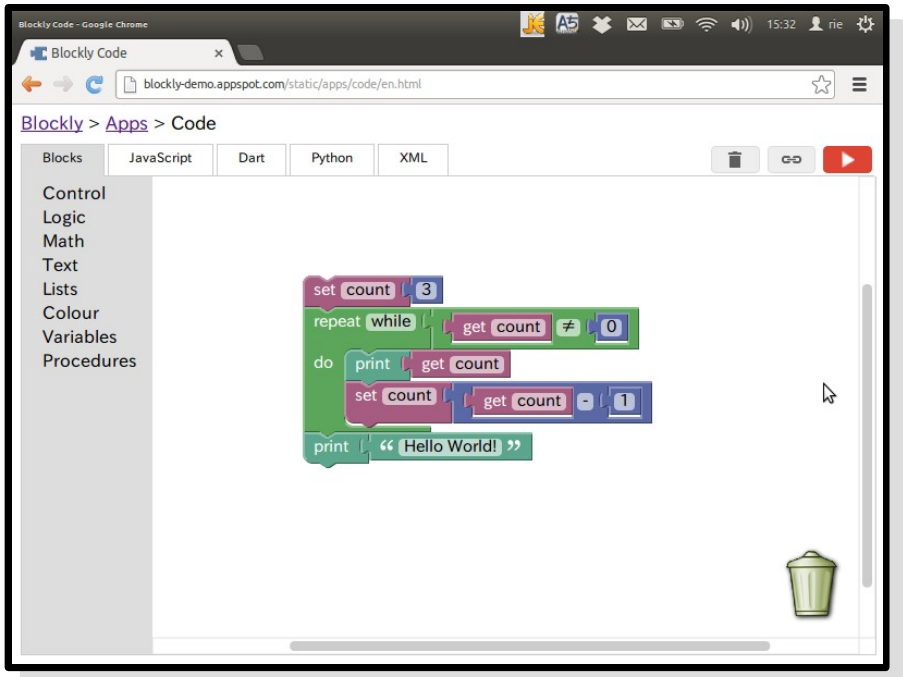
Blockly サイト
https://code.google.com/p/blockly/

Blockly紹介ビデオ（画質が粗くてごめんなさい）
http://www.youtube.com/watch?v=p_HM3XD4-QM&feature=youtu.be

Blockly について



Blockly サイト



Blockly ビジュアル・エディタ

Blockly の仲間たち

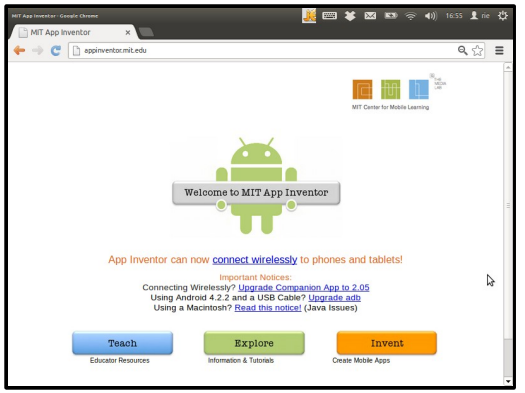
Blocklyは、あまた存在する
ビジュアル・プログラミング環境の１つです。

ビジュアル・プログラミング環境は、
世界中で今も誕生し続けています。

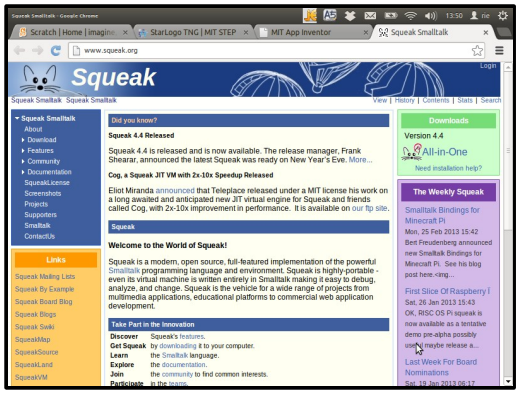
興味をもたれた方は、
Google 検索されては
いかがでしょうか。

Blockly の仲間たち

SCRATCH
http://scratch.mit.edu/



AppInventor
http://appinventor.mit.edu/



Squeak
http://www.squeak.org/

StarLogoプロジェクト
http://education.mit.edu/projects/starlogo-tng

文部科学省プログラミン
http://www.mext.go.jp/programin/

Blockly の特徴

- Webブラウザのみで利用可能です。
  開発環境のダウンロードやプラグインの追加も
  必要ありません。（サーバすら不要です）

- 各国語対応も考慮されています。
  ブロック表示メッセージのリソース化手法の提供
  のみならず、右から左に流れる言語でも対応できる
  ように設計されています。

  ※メッセージ表示は、リソース化を行わなくても(直書でも)可能です。

Blockly の特徴

- プログラム言語を問わないコード・テキスト生成

  任意のプログラミング言語のコード・テキスト生成が可能となるよう設計されています。
  自由にサポート言語が追加開発できるのです。

  公開CODEサンプルでは、JavaScriptやPythonのコードがエクスポートできるようになっています。

  ※2013/02までは、Dartコードのエクスポート・サポートも行われていました。

Blockly の特徴

・独自拡張も自分のサイトでの使用も自由です。

Blocklyは、Apache License 2.0ライセンスのオープンソース・プロジェクトです。

独自拡張したソースの頒布は、派生成果物に
Apache License 2.0 のコピー(テキスト)と、
参照元の帰属告知のコピー(派生成果物に無関係物は除く)と、
独自変更(拡張)部の告知を含めれば、自由に行えます。
(詳しくは、脚注を御参照ください)

参考先

Apache License 2.0
http://www.apache.org/licenses/LICENSE-2.0

Apache License, Version 2.0 (日本語訳)
http://sourceforge.jp/projects/opensource/wiki/licenses%2FApache_License_2.0

Blockly の特徴

・教育環境での利用のみにとどまりません。

Blocklyは、
任意プログラム言語のコード・テキスト生成や、
任意機能を表すカスタム・ブロック作成が、
独自に開発できるよう設計されています。

つまり特定ハード用のプログラム開発や、
業務用DSLの独自開発にも応用可能なのです。

参考先

紹介しました特徴については、下記の Blockly wiki ページを参考にいたしました。

Alternatives
https://code.google.com/p/blockly/wiki/Alternatives

Blockly 独自拡張の例

Arudino ビジュアルプログラミング・エディタ

BlocklyDuino

Blocklyを独自拡張して、
Arudino用の
ビジュアル・プログラミング・エディタ
BlocklyDuinoを作成された方もおられます。

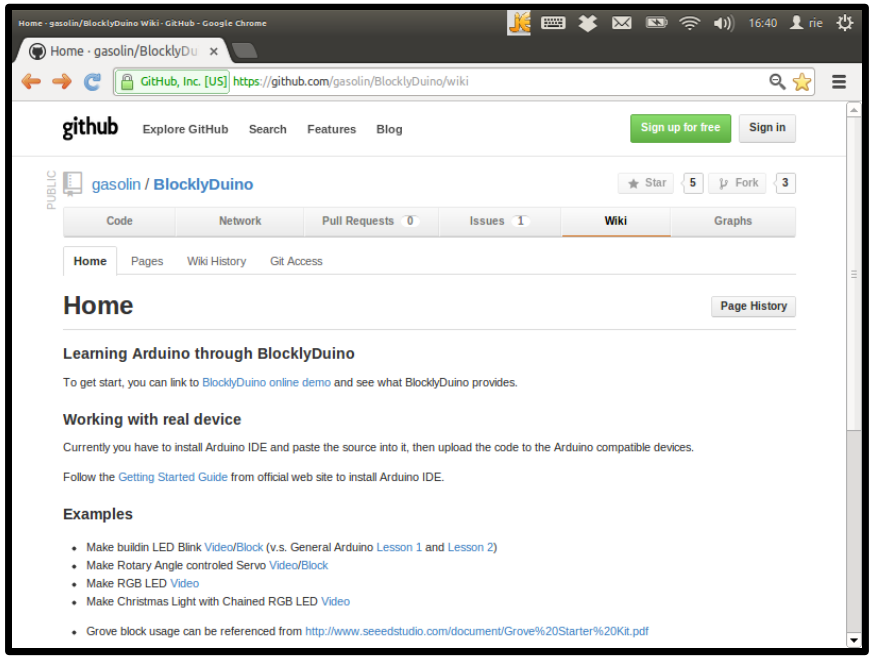
gasolin / BlocklyDuino
https://github.com/gasolin/BlocklyDuino/wiki

BlocklyDuino visual editor demo
http://www.gasolin.idv.tw/public/blockly/demos/blocklyduino/index.html

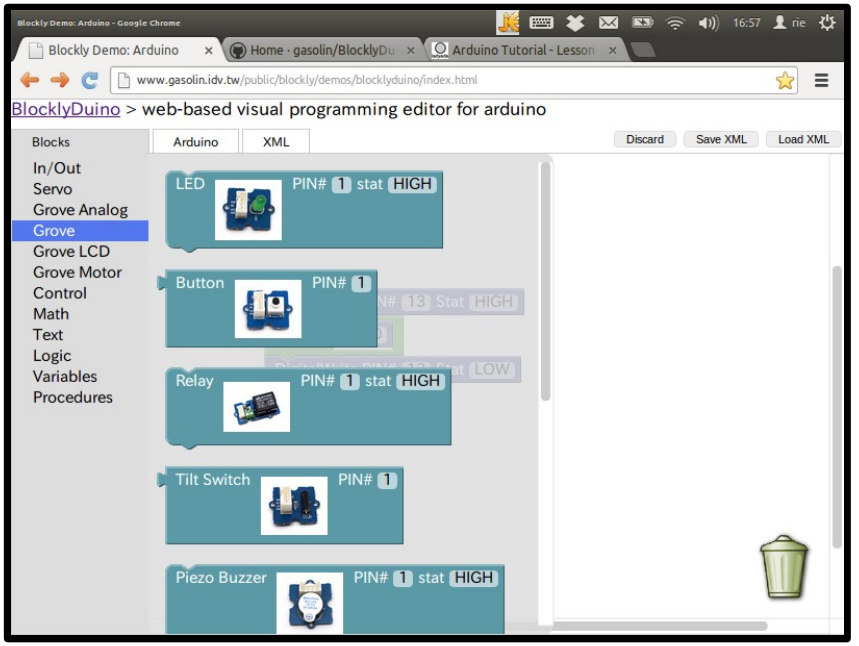
Blockly 独自拡張の例

Arudino ビジュアルプログラミング・エディタ

BlocklyDuino



BlocklyDuino サイト

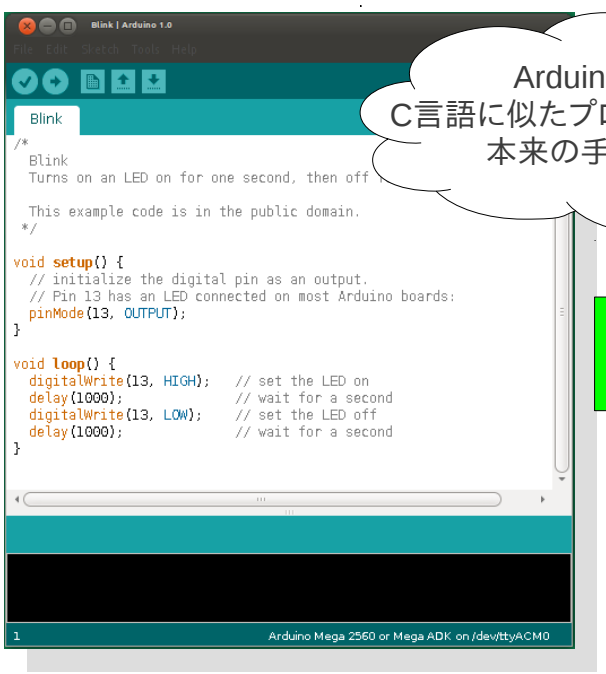


BlocklyDuino エディタ･デモ

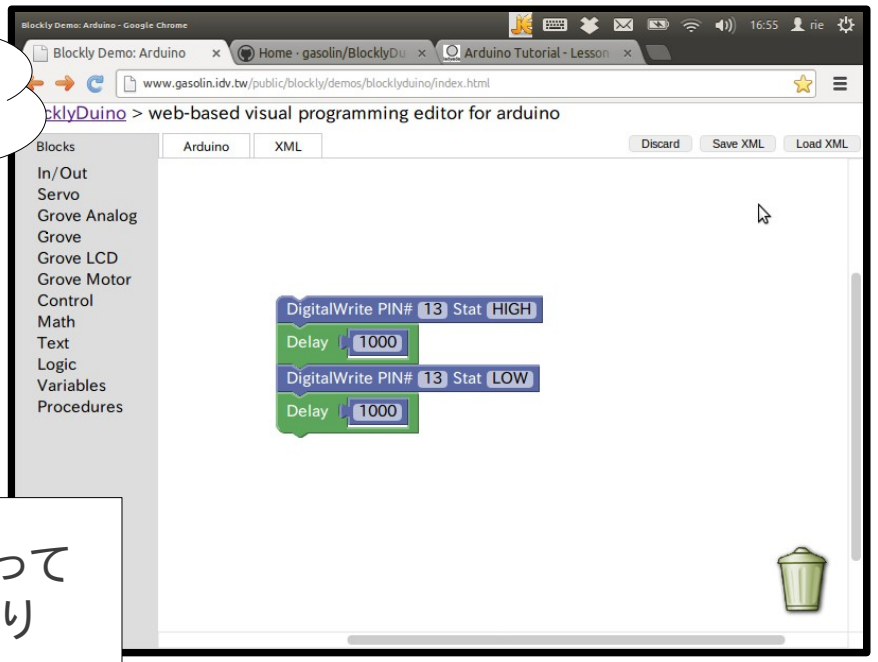
Blockly 独自拡張の例

Arudino ビジュアルプログラミング・エディタ

BlocklyDuino



Arduino開発は、
C言語に似たプログラミング言語が
本来の手法ですが…



Lチカだって
この通り

ArudinoIDE

BlocklyDuino エディタ･デモ

制限事項

- ローカルファイルでは、エラーとなる場合がある。

  Chromeブラウザでは、セキュリティ上の制限のため file://スキーマ利用時のインライン・フレームのテンプレート展開に失敗します。(issue 47416)
  http://code.google.com/p/chromium/issues/detail?id=47416

  このため Blockly プロジェクトから取得したソースコードのAppsコンテンツが実行できません。

制限事項

・Chromeでは、httpサーバを介した利用を勧めます。

独自開発のためローカルファイルを利用する場合は、ローカル http サーバを立ち上げるか、FirefoxやSafariやOperaブラウザの御利用を検討ください。

【補足】Blocklyプロジェクト・ソースコードのDemosは、iframeやテンプレート未使用ですChoromeであってもローカルファイル(file://スキーマ)からの実行が可能です。

制限事項

・ファイルからのインポートに対応していません。

コード・テキストやブロック構造XMLの
直接ファイル保存や読込は、対応していません。
作成プログラムは、エクスポートのみ可能です。

Blockly > Apps > Codeで作成したプログラムは、
ブロック構造のXMLテキストエリア内容をコピーして
テキストエディタに貼りつけファイル保管すれば、
保管済みのXML内容を再びXMLテキストエリアに
貼り付けることで再現することができます。

制限事項

- 高機能ブロックは、提供していません。
  言語仕様を一般化 (幅広いプログラム言語への対応思想より) するため、
  提供ブロックの機能は、低レベルとなっています。
  特定の処理を１ブロックで実現するには、独自に
  カスタム・ブロックを開発する必要があります。
- 大規模プログラム開発を対象としていません。
  比較的小規模なプログラミングを対象とした
  設計となっています。

参考先

制限事項については、下記の Blockly wiki ページを参考にいたしました。

FAQ - High Level
https://code.google.com/p/blockly/wiki/FAQHighLevel

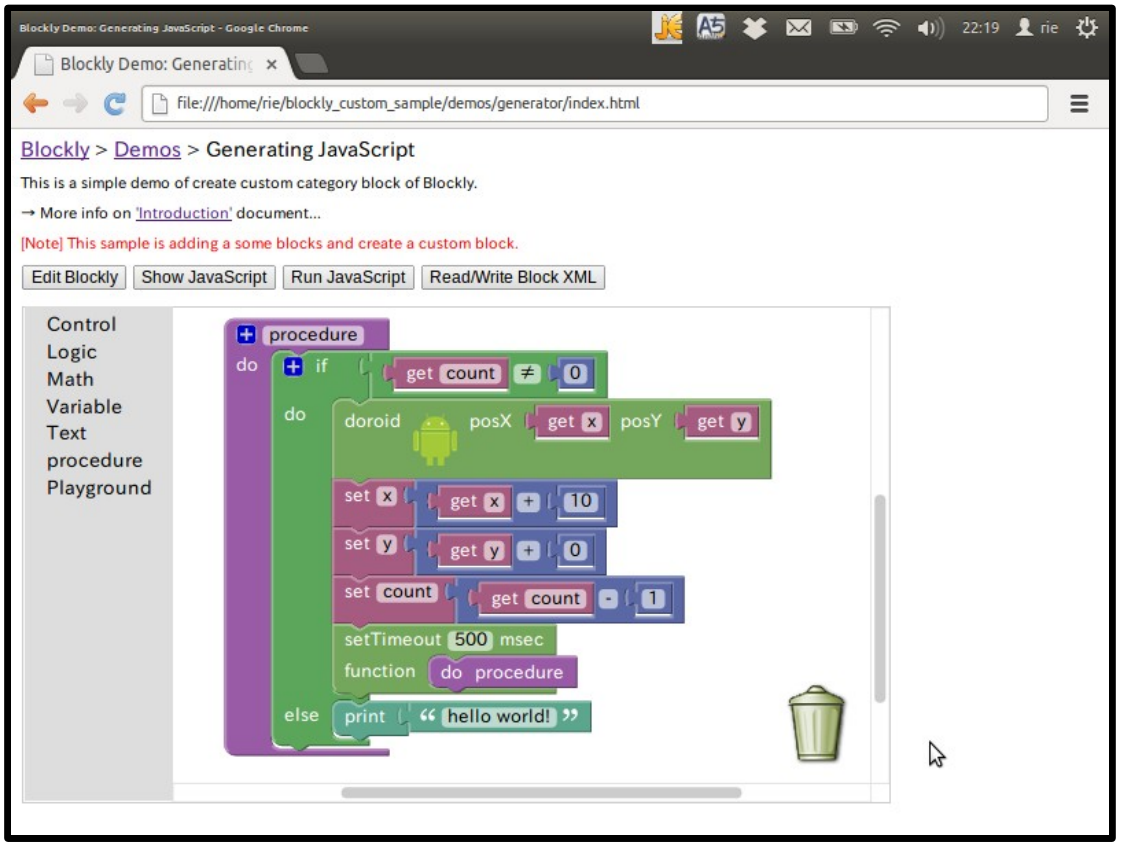
Blockly紹介編と次章について

Blockly についての一般的な説明は、以上です。

聴講者(読者)がプログラミング知識があるとして、
エディタやブロックの使い方は、省いています。
(お使いになれば、すぐに判ると思います)

以降から、
簡単なカスタム・ビジュアルエディタ作成を介して
Blocklyの独自拡張について説明いたします。

Blockly紹介編と次章について



サンプル作成する簡易カスタム･ビジュアルエディタの外観

Blockly独自拡張の入門書

- Introduction(入門書)ページ

  基本的な実装(APIの利用や解説)については、
  Introduction(入門書)ページを始まりとして、
  一連のwikiページを辿ることで学びます。

  正式な技術情報源は、上記ページ(とML)のみです。
  サンプルの作成においても、上記ページの情報と
  ソースコード解析のみで対応しました。

参考先

Introduction
https://code.google.com/p/blockly/wiki/Installation

Blcockly wiki page
https://code.google.com/p/blockly/w/list

Blockly独自拡張の入門書

- Introduction(入門書)は必読です。

  Introduction(入門書)の一連のページは、
  英語ドキュメントですが、30ページもありません。
  独自拡張を行いたい方は、必ず目を通してください。

  正式な技術的情報は、上記を読解するのみなため
  私からの説明は、サンプル作成にあたって感じた
  キーポイントに絞らせていただきます。

ソースコード取得

・ソースコード取得には、SVNを利用します。

開発メンバーでない方は、下記のSVNコマンドで、
SVNサーバからHTTPを介した、リードオンリーの
ワークコピーを匿名でチェックアウトして下さい。

```
svn checkout http://blockly.googlecode.com/svn/trunk/ blockly-read-only
```

eclipseなどのSVNクライアントを用いれば、コマンドを使わずに
上記URLリポジトリから匿名でソースコード一式が取得できます。

参考先

ソースコード取得説明
https://code.google.com/p/blockly/source/checkout

【注意】Zipファイルでのソースコード提供サービスは未対応です。

ソースコード取得【補足】

・正式開発では、コンパイルを行っています。

Blocklyは、JavaScriptで動作しているので、
ソースコードを改修しての直接実行も可能ですが、
開発要件的には、コア部のソースを圧縮するなど、
コンパイルが必要になっています。

ソースコードのルートには、
Makeファイルも存在しますが、
make環境構築について調べ切れていません。
(ごめんなさい)

ソースコード取得【補足】

・コントリビュートについて

コントリビュートするに当たっては、
dept_toolsという、チェックアウトや
コードレビューを管理するために呼び出される
スクリプト・パッケージの使用が必要だそうです。

詳しくは、以下のWikiページを御参照ください。

How to contribute to the blockly project
https://code.google.com/p/blockly/wiki/ContributingCode

ソースコード取得【補足】

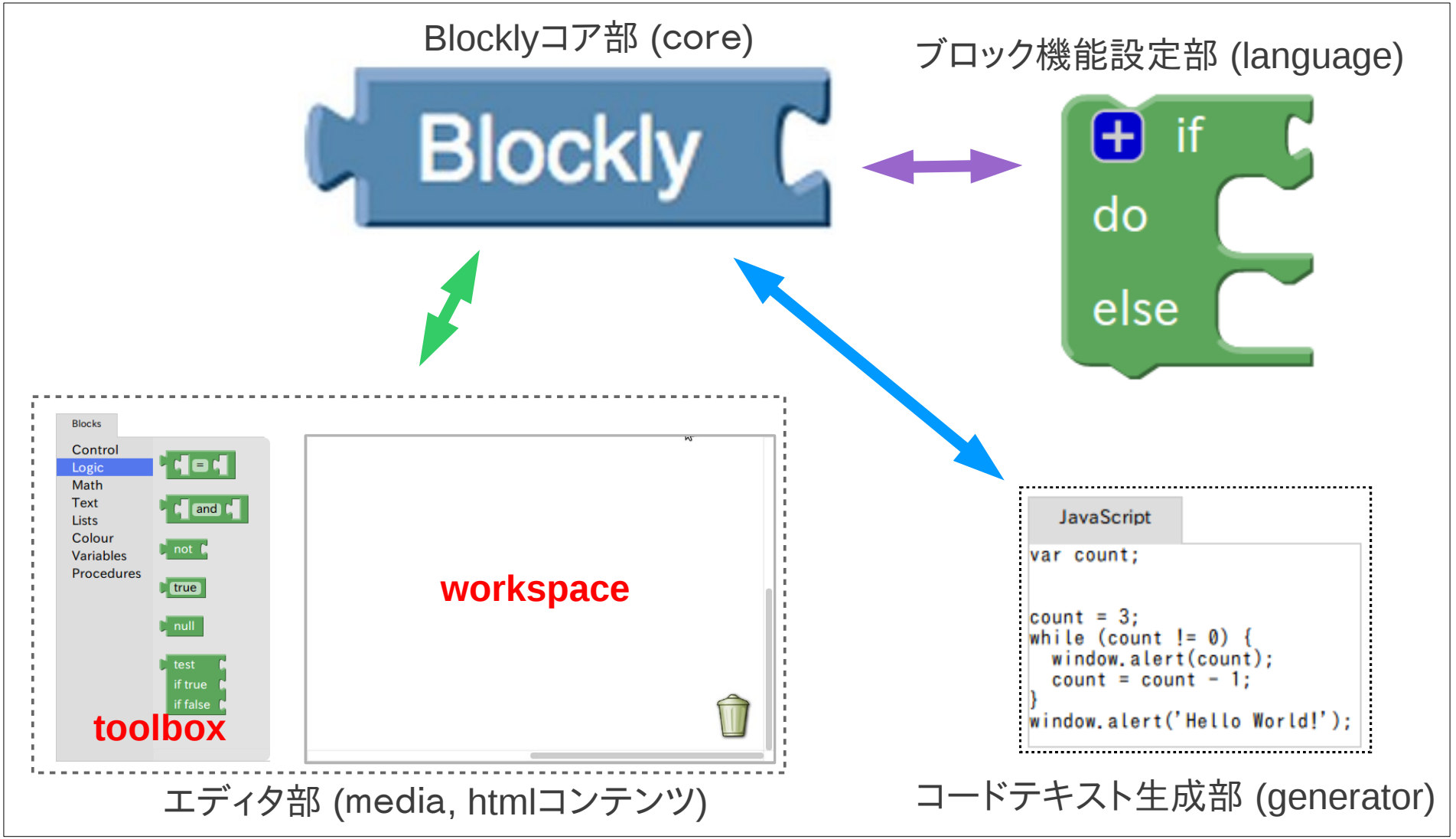
- Blocklyソース･ツリー

  取得したソースコード一式のフォルダ構成は、
  以下のようになっていました。

```
<Blocklyソース>/core      …  Blocklyコア機能・モジュール          Blockly全体の基本機能部
                language  …  ブロック機能設定・モジュール      ブロック毎の形状や役割の設定部
                              ブロック機能を担うcommon、各国語
                              (en,…zh_tw)のサブフォルダを含む
                generator …  コードテキスト生成・モジュール  ブロック毎のコードテキスト生成部
                              プログラム言語ごとのサブフォルダ
                              (javascript、python、dart)を含む
                media     …  エディタのリソース・モジュール          エディタ部のリソース
                apps      …  Blocklyでできることの説明用アプリ
                demos     …  Blocklyの機能紹介用サンプル          エディタ部のHTML/JSも含まれる
                appengine …  Cloud Strage用のモジュール           エディタ部のHTML/JSも含まれる
                externs   …  詳細不明
                tests     …  詳細不明(テストツール？)
```

ソースルートには、Blocklyコア機能全体を圧縮した
blockly_compressed.js が配置されています。

Blockly全体構成イメージ



作成サンプルについて

Blockly独自拡張のサンプルは、ソースコードの
demos/generatorを元に作成しています。

サンプルのソースコード一式は、
blockly_custom_sampleフォルダに収めました。
改造内容が読み易いよう、実行に必要な最少限のファイル
のみを配置し、改修部前後にコメントを入れたほか、
実装内容についてのコメントも入れました。

以降の説明では、
サンプルのソース内容と見比べて読み進めて下さい。

作成サンプルについて

・新規作成および改修ファイルのみの一覧

```
◎/blockly_compressed.js                  … (改)圧縮の展開のみ行っています。
  |
  /demos/index.html                      … (改)サンプルへの導線改修のみ
  |      |
  |      /generator/index.html           … (改)エディタ部の設定処理部
  |                 |                      toolboxで利用するブロック・カテゴリの設定を含む
  |                 /droid.png           … (新)ドロイド君画像(画面表示用リソース)
  |
  /generators/javascript.js              … (改)JavaScriptコードテキスト生成の全体処理部
  |           |                            コード生成の初期化時や終了時の処理を含む
  |           |
  |           /javascript/playground.js  … (新)playgroundカテゴリの
  |                                        JavaScriptコードテキスト生成部
  |
  /language/common/playground.js         … (新)playgroundカテゴリのブロック機能設定部
           |
           /en/_messages.js              … (改)ブロックに表示する文言設定(英語版)
```

拡張子が org のファイルは、改造前のオリジナル・ファイルです。

作成サンプルについて

・独自拡張部についての説明

エディタ部の拡張
toolboxで利用するブロック・カテゴリの追加と、
プログラム実行時の画面(playground 遊戯場)や、
JavaScriptのコードテキストとXMLのテキストを
workspace内容(プログラム内容→ブロック構造)と
同期させて表示する画面を追加しています。

ここでは、toolbox内容の設定や、
workspace内容とテキストやHTML(DOM)
オブジェクトを同期させる実装を行っています。

作成サンプルについて

・独自拡張部についての説明

ブロック機能設定とコードテキスト生成の追加

新規追加のカスタム・ブロックとして、

新規追加のカスタム・ブロックとして、プログラム実行時の
画面と表示物を表す playgroundという「ブロック・カテゴリ」
(JSファイル)をlanguageとgeneratorsフォルダに新設して、
下記３つの「ブロック項目」をカテゴリ内に設けることで、
対応する３つのカスタム・ブロックを新規作成しています。

```
playground        … JavaScript実行時の画面を表すブロック
playground_droid … ドロイド君を画面に表示するブロック
playground_wait   … 処理を指定時間待機させるブロック
```


エディタ部の実装内容

- toolboxで利用するブロックに関する実装

  新規ブロック・カテゴリ playground の
  ブロック(各ブロック項目)を利用する場合(part1)

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
  head 部

実装内容：

```
<script type="text/javascript" src="../../blockly_compressed.js"></script>
～ 省略 ～
<script type="text/javascript" src="../../language/en/_messages.js"></script>
～ 省略 ～
<script type="text/javascript" src="../../language/common/playground.js"></script>
～ 省略 ～
<script type="text/javascript" src="../../generators/javascript.js"></script>
～ 省略 ～
<script type="text/javascript" src="../../generators/javascript/playground.js"></script>
```

【参照】Installation: Language Generators ← generators部の実装に関係します
https://code.google.com/p/blockly/wiki/LanguageGenerators

エディタ部の実装内容

- toolboxで利用するブロックに関する実装

  新規ブロック・カテゴリ playground の
  ブロック(各ブロック項目)を利用する場合(part2)

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
  body部

実装内容：

```
<xml id="toolbox" style="display: none">
  ～ 省略 ～
  <category name="Playground">
    <block type="playground"></block>
    <block type="playground_doroid"></block>
    <block type="playground_wait"></block>
  </category>
  ～ 省略 ～
  </xml>
```

【参照】Installation: Defining the Toolbox
https://code.google.com/p/blockly/wiki/Toolbox

エディタ部の実装内容

- workspaceのブロック構造とXMLの相互同期

  workspaceのブロック構造と、
  XMLテキストを相互に同期させる実装(part1)

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
scriptタグ・ブロック内

実装内容：（ブロック構造→XMLテキスト）

```
 function readXml() {
      ～ 省略 ～
      var xmlDom = Blockly.Xml.workspaceToDom(Blockly.mainWorkspace);
      ～ 省略 ～
       xmlText = Blockly.Xml.domToPrettyText(xmlDom); //XML テキストのインデント整形を行います。
      ～ 省略 ～
 }
```

【参照】Introduction (Importing and Exporting Blocks)
https://code.google.com/p/blockly/wiki/Installation

エディタ部の実装内容

- workspaceのブロック構造とXMLの相互同期

  workspaceのブロック構造と、
  XMLテキストを相互に同期させる実装(part2)

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
scriptタグ・ブロック内

実装内容：（XMLテキスト→ブロック構造）

```
function switchDisplay(selectorId) {
  ～ 省略 ～
     var xmlDom = Blockly.Xml.textToDom(xmlText);
  ～ 省略 ～
     Blockly.mainWorkspace.clear();
     Blockly.Xml.domToWorkspace(Blockly.mainWorkspace, xmlDom);
  ～ 省略 ～
}
```


エディタ部の実装内容

- workspaceからJavaScriptコードテキスト生成

  workspaceのブロック構造から、
  JavaScriptのコードテキストを生成する実装

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
scriptタグ・ブロック内

実装内容：（ブロック構造→JavaScriptコードテキスト）

```
function showCode() {
  var code = Blockly.Generator.workspaceToCode('JavaScript');
}
```


エディタ部の実装内容

- workspace再描画
  workspace内容を再描画させる実装

対象ソース・ファイル：
<サンプル・フォルダ>/demos/generator/index.html

実装場所：
scriptタグ・ブロック内

実装内容：

```
function editBlock() {
  Blockly.mainWorkspace.renderBlocks();
}
```


ブロック機能設定の実装

- ブロック playground_doroid の機能設定
  新規ブロック項目 playground_doroid の
  ブロック機能(形状含む)設定実装 (part1)

対象ソース・ファイル：
<サンプル・フォルダ>/language/common/playground.js

実装内容：

```
// playground(遊び場)画面の doroido(アイコン表示)ブロック設定
Blockly.Language.playground_doroid = {
  ～ 省略 ～
  init: function() {
    //ブロック色設定(引数はHSV色)
    this.setColour(100);
```

【参照】Creating Custom Blocks: Defining Blocks
https://code.google.com/p/blockly/wiki/DefiningBlocks
part1からpart3までの一連の処理は、上記ドキュメントに関連しています。

【参照】Creating Custom Blocks: Defining a Block: Creating Mutators
https://code.google.com/p/blockly/wiki/CreatingMutators
ここでは利用していませんが、ブロック形状のユーザ変更も参照して下さい。

ブロック機能設定の実装

- ブロック playground_doroid の機能設定
  新規ブロック項目 playground_doroid の
  ブロック機能(形状含む)設定実装 (part2)

実装内容：

```
//作業変数の値を設定する、入力フィールド(リストボックス等)を追加
//【注】入力フィールドは、必須ではありません。(指定しなくてもでも良い)
this.appendDummyInput()
    .appendTitle(Blockly.LANG_PLAYGROUND_DOROID_INPUT_ICON)
    .appendTitle(new Blockly.FieldImage('doroid.png', 36, 44));
//作業変数の値を設定する、入力ブロック行を追加
this.appendValueInput('POS_X')
    .setCheck(Number)
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendTitle(Blockly.LANG_PLAYGROUND_DOROID_INPUT_X);
//作業変数の値を設定する、入力ブロック行を追加
this.appendValueInput('POS_Y')
    .setCheck(Number)
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendTitle(Blockly.LANG_PLAYGROUND_DOROID_INPUT_Y);
```


ブロック機能設定の実装

- ブロック playground_doroid の機能設定
  新規ブロック項目 playground_doroid の
  ブロック機能(形状含む)設定実装 (part3)

実装内容：

```
    //前ブロック受付指定(有効)
    this.setPreviousStatement(true);
    //前ブロック受付指定(有効)
    this.setNextStatement(true);
    //入力項目のインライン指定(あり)
    //【注】入力フィールドと入力ブロックが複数ある場合、
    //      true 指定により、それらを１行に表示します。
    this.setInputsInline(true);
    //ツールチップ表示指定
    // Assign 'this' to a variable for use in the tooltip closure below.
    var thisBlock = this;
    this.setTooltip(function() {
      return Blockly.LANG_PLAYGROUND_DOROID_TOOLTIP.replace('%1',
          thisBlock.getTitleValue('VAR'));
    });
  },
  ～ 省略 ～
};
```


コードテキスト生成の実装

- ブロック playground_doroid のコード生成
  新規ブロック項目 playground_doroid の
  JavaScriptコードテキスト生成実装（part1）

対象ソース・ファイル：
<サンプル・フォルダ>/generators/javascript/playground.js

実装内容：（ブロック入力値の取得部）

```
// playground(遊び場)の doroid(アイコン表示物)設定
Blockly.JavaScript.playground_doroid = function() {
  // ドロイドくんアイコン設定
  var argument0 = Blockly.JavaScript.valueToCode(this, 'POS_X',
      Blockly.JavaScript.ORDER_ASSIGNMENT) || '0';
  var argument1 = Blockly.JavaScript.valueToCode(this, 'POS_Y',
      Blockly.JavaScript.ORDER_ASSIGNMENT) || '0';
  ～省略～
```

【参照】Creating Custom Blocks: Generating Code: Operator Precedence（演算を伴う可能性のある値の優先順位指定）
https://code.google.com/p/blockly/wiki/OperatorPrecedence

【参照】Creating Custom Blocks: Generating Code
https://code.google.com/p/blockly/wiki/GeneratingCode
ここでは利用していませんが、ブロック入力値の取得について上記も参照して下さい。

コードテキスト生成の実装

- ブロック playground_doroid のコード生成
  新規ブロック項目 playground_doroid の
  JavaScriptコードテキスト生成実装（part2）

実装内容：（生成JavaScriptコードテキスト設定部）

```
  var code;
  code = "if (_playgroundCanvasVar_) {¥n"
       + "  var _pg_context_ = _playgroundCanvasVar_.getContext('2d');¥n"
       + "  _pg_context_.fillStyle = _playgroundCanvasVar_.style['background-color'];¥n"
       + "  _pg_context_.fillRect(_doroidImageX_,_doroidImageY_, 36,44);¥n"
       + "  _doroidImageX_ = " + argument0 + ";¥n"
       + "  _doroidImageY_ = " + argument1 + ";¥n"
       + "  _pg_context_.drawImage(_doroidImageVar_, _doroidImageX_,_doroidImageY_, 36,44);¥n"
       + "}¥n";
  return code;
};
```


コードテキスト生成の実装

・多重記述抑止が必要なコード生成

Blocilyには、変数の多重宣言抑止機構が、
予め提供済となっています。

ですが独自拡張では、当然自分での対応になります。
このため、プログラム全体の初期処理と終了処理に
パッチをあて多重記述を抑止するフラグの初期化と、
実行済時の初期化コード対応処理を組み込みました。
(この手法は、ダーティハックに相当します)

【参照】Creating Custom Blocks: Generating Code: Caching Arguments（変数の多重宣言抑止記述あり）
https://code.google.com/p/blockly/wiki/CachingArguments

コードテキスト生成の実装

- 多重記述抑止が必要なコード生成

実行時のフラグON処理と
多重記述を抑止するフラグの定義および初期化コードの定義

対象ソース・ファイル：
<サンプル・フォルダ>/generators/javascript/playground.js

実装内容：（実行済フラグ設定と、初期化コード指定）

```
// playground(遊び場)の doroid(アイコン表示物)設定
Blockly.JavaScript.playground_doroid = function() {
  ～省略～
  Blockly.JavaScript.playground_doroid.isExist = true;
  ～省略～
};

// doroid 定義確認フラグ
Blockly.JavaScript.playground_doroid.isExist = null;
// doroid 初期設定コード
Blockly.JavaScript.playground_doroid.initCode =
    "var _doroidImageVar_ = new Image();¥n" + "_doroidImageVar_.src = 'doroid.png';¥n"
  + "var _doroidImageX_ = 0;¥n" + "var _doroidImageY_ = 0;";
```


コードテキスト生成の実装

- 多重記述抑止が必要なコード生成

プログラム全体初期化処理での
多重記述抑止フラグの強制初期化対応

対象ソース・ファイル：
<サンプル・フォルダ>/generators/javascript.js

実装内容：（実行済フラグの強制初期化）

```
Blockly.JavaScript.init = function() {
～省略～
//2013/03/08 my add line start
  //playground 利用フラグを強制初期化
  Blockly.JavaScript.playground.isExist = false;
  //doroid_doroid 利用フラグを強制初期化
  Blockly.JavaScript.playground_doroid.isExist = false;
  //doroid_wait 利用フラグを強制初期化
  Blockly.JavaScript.playground_wait.isExist = false;
//2013/03/08 my add line end
};
```


コードテキスト生成の実装

・多重記述抑止が必要なコード生成

プログラム全体終了処理での
多重記述抑止フラグON時の初期化コード強制追加対応

対象ソース・ファイル：
<サンプル・フォルダ>/generators/javascript.js

実装内容：（実行済時の初期化コード強制追加）

```
Blockly.JavaScript.finish = function(code) {
～省略～
//2013/03/08 my add line start
  //playground が利用されていれば、definitions(定義物)に playground の設定を強制追加
  if(Blockly.JavaScript.playground.isExist === true){
    Blockly.JavaScript.definitions_['playground'] = Blockly.JavaScript.playground.initCode;
  }
  //playground_doroid が利用されていれば、definitions(定義物)に playground_doroid の設定を強制追加
  if(Blockly.JavaScript.playground_doroid.isExist === true){
    Blockly.JavaScript.definitions_['playground_doroid'] = Blockly.JavaScript.playground_doroid.initCode;
  }
//2013/03/08 my add line end
～省略～
};
```


サンプル･エディタの実行

サンプル作成しました
カスタム・ビジュアルエディタ実装は、
以上のようになっています。

簡単な拡張ですが、カスタム・ブロック追加や
XMLテキストの貼付によるプログラミング済みの
ブロック構造の再現もできるようになっています。

プログラム済みのサンプルXMLを張り付けて、
どのような動きをするのか確認してみましょう。

サンプル･エディタの実行

サンプル一式が含まれたフォルダ直下の
droid_hello_world.xml ファイルを
テキストエディタで開いてください。

そして全テキストをコピーして
[Read/Write Block XML]をクリック、
XMLテキストエリアにコピーを貼付てから、
[Run JavaScript]をクリックしてみて下さい…

サンプル･エディタの実行

…いかがでしたか？

サンプル内容は、つまらないものですが、
独自拡張ブロックも下手なりに作れたのです。

「Blocklyの独自拡張は、手の届かないものではない」
と思ってくださいましたら幸です。

Google Blockly で
ビジュアルプログラムを楽しむ

ご清聴、ありがとうございました。