```
/**
 * Blockly Games: Maze Blocks
 *
 * Copyright 2012 Google Inc.
 * https://github.com/google/blockly-games
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/**
 * @fileoverview Blocks for Blockly's Maze application.
 * @author fraser@google.com (Neil Fraser)
 */
'use strict';

goog.provide('Maze.Blocks');

goog.require('Blockly');
goog.require('Blockly.JavaScript');
goog.require('BlocklyGames');


/**
 * Common HSV hue for all movement blocks.
 */
Maze.Blocks.MOVEMENT_HUE = 290;

/**
 * HSV hue for loop block.
 */
Maze.Blocks.LOOPS_HUE = 120;

/**
 * Common HSV hue for all logic blocks.
 */
Maze.Blocks.LOGIC_HUE = 210;

/**
 * Left turn arrow to be appended to messages.
 */
Maze.Blocks.LEFT_TURN = ' \u21BA';

/**
 * Left turn arrow to be appended to messages.
 */
Maze.Blocks.RIGHT_TURN = ' \u21BB';

// Extensions to Blockly's language and JavaScript generator.

Blockly.Blocks['maze_moveForward'] = {
  /**
   * Block for moving forward.
   * @this Blockly.Block
   */
  init: function() {
    this.jsonInit({
      "message0": BlocklyGames.getMsg('Maze_moveForward'),
      "previousStatement": null,
      "nextStatement": null,
```

```
70        "colour": Maze.Blocks.MOVEMENT_HUE,
71        "tooltip": BlocklyGames.getMsg('Maze_moveForwardTooltip')
72      });
73    }
74  };
75
76  Blockly.JavaScript['maze_moveForward'] = function(block) {
77    // Generate JavaScript for moving forward.
78    return 'moveForward(\'block_id_' + block.id + '\');\n';
79  };
80
81  Blockly.Blocks['maze_turn'] = {
82    /**
83     * Block for turning left or right.
84     * @this Blockly.Block
85     */
86    init: function() {
87      var DIRECTIONS =
88          [[BlocklyGames.getMsg('Maze_turnLeft'), 'turnLeft'],
89           [BlocklyGames.getMsg('Maze_turnRight'), 'turnRight']];
90      // Append arrows to direction messages.
91      DIRECTIONS[0][0] += Maze.Blocks.LEFT_TURN;
92      DIRECTIONS[1][0] += Maze.Blocks.RIGHT_TURN;
93      this.setColour(Maze.Blocks.MOVEMENT_HUE);
94      this.appendDummyInput()
95          .appendField(new Blockly.FieldDropdown(DIRECTIONS), 'DIR');
96      this.setPreviousStatement(true);
97      this.setNextStatement(true);
98      this.setTooltip(BlocklyGames.getMsg('Maze_turnTooltip'));
99    }
100 };
101
102 Blockly.JavaScript['maze_turn'] = function(block) {
103   // Generate JavaScript for turning left or right.
104   var dir = block.getFieldValue('DIR');
105   return dir + '(\'block_id_' + block.id + '\');\n';
106 };
107
108 Blockly.Blocks['maze_if'] = {
109   /**
110    * Block for 'if' conditional if there is a path.
111    * @this Blockly.Block
112    */
113   init: function() {
114     var DIRECTIONS =
115         [[BlocklyGames.getMsg('Maze_pathAhead'), 'isPathForward'],
116          [BlocklyGames.getMsg('Maze_pathLeft'), 'isPathLeft'],
117          [BlocklyGames.getMsg('Maze_pathRight'), 'isPathRight']];
118     // Append arrows to direction messages.
119     DIRECTIONS[1][0] += Maze.Blocks.LEFT_TURN;
120     DIRECTIONS[2][0] += Maze.Blocks.RIGHT_TURN;
121     this.setColour(Maze.Blocks.LOGIC_HUE);
122     this.appendDummyInput()
123         .appendField(new Blockly.FieldDropdown(DIRECTIONS), 'DIR');
124     this.appendStatementInput('DO')
125         .appendField(BlocklyGames.getMsg('Maze_doCode'));
126     this.setTooltip(BlocklyGames.getMsg('Maze_ifTooltip'));
127     this.setPreviousStatement(true);
128     this.setNextStatement(true);
129   }
130 };
131
132 Blockly.JavaScript['maze_if'] = function(block) {
133   // Generate JavaScript for 'if' conditional if there is a path.
134   var argument = block.getFieldValue('DIR') +
135       '(\'block_id_' + block.id + '\')';
136   var branch = Blockly.JavaScript.statementToCode(block, 'DO');
137   var code = 'if (' + argument + ') {\n' + branch + '}\n';
138   return code;
```

```
};

Blockly.Blocks['maze_ifElse'] = {
  /**
   * Block for 'if/else' conditional if there is a path.
   * @this Blockly.Block
   */
  init: function() {
    var DIRECTIONS =
        [[BlocklyGames.getMsg('Maze_pathAhead'), 'isPathForward'],
         [BlocklyGames.getMsg('Maze_pathLeft'), 'isPathLeft'],
         [BlocklyGames.getMsg('Maze_pathRight'), 'isPathRight']];
    // Append arrows to direction messages.
    DIRECTIONS[1][0] += Maze.Blocks.LEFT_TURN;
    DIRECTIONS[2][0] += Maze.Blocks.RIGHT_TURN;
    this.setColour(Maze.Blocks.LOGIC_HUE);
    this.appendDummyInput()
        .appendField(new Blockly.FieldDropdown(DIRECTIONS), 'DIR');
    this.appendStatementInput('DO')
        .appendField(BlocklyGames.getMsg('Maze_doCode'));
    this.appendStatementInput('ELSE')
        .appendField(BlocklyGames.getMsg('Maze_elseCode'));
    this.setTooltip(BlocklyGames.getMsg('Maze_ifelseTooltip'));
    this.setPreviousStatement(true);
    this.setNextStatement(true);
  }
};

Blockly.JavaScript['maze_ifElse'] = function(block) {
  // Generate JavaScript for 'if/else' conditional if there is a path.
  var argument = block.getFieldValue('DIR') +
      '(\'block_id_' + block.id + '\')';
  var branch0 = Blockly.JavaScript.statementToCode(block, 'DO');
  var branch1 = Blockly.JavaScript.statementToCode(block, 'ELSE');
  var code = 'if (' + argument + ') {\n' + branch0 +
             '} else {\n' + branch1 + '}\n';
  return code;
};

Blockly.Blocks['maze_forever'] = {
  /**
   * Block for repeat loop.
   * @this Blockly.Block
   */
  init: function() {
    this.setColour(Maze.Blocks.LOOPS_HUE);
    this.appendDummyInput()
        .appendField(BlocklyGames.getMsg('Maze_repeatUntil'))
        .appendField(new Blockly.FieldImage(Maze.SKIN.marker, 12, 16));
    this.appendStatementInput('DO')
        .appendField(BlocklyGames.getMsg('Maze_doCode'));
    this.setPreviousStatement(true);
    this.setTooltip(BlocklyGames.getMsg('Maze_whileTooltip'));
  }
};

Blockly.JavaScript['maze_forever'] = function(block) {
  // Generate JavaScript for repeat loop.
  var branch = Blockly.JavaScript.statementToCode(block, 'DO');
  if (Blockly.JavaScript.INFINITE_LOOP_TRAP) {
    branch = Blockly.JavaScript.INFINITE_LOOP_TRAP.replace(/%1/g,
        '\'block_id_' + block.id + '\'') + branch;
  }
  return 'while (notDone()) {\n' + branch + '}\n';
};
```